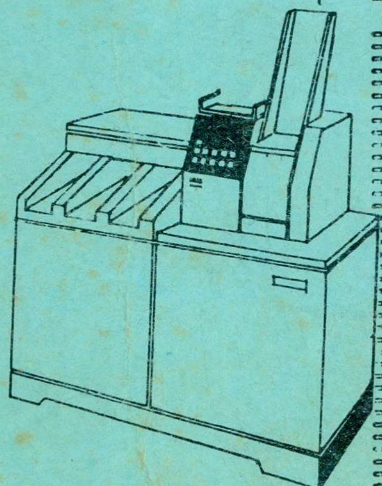
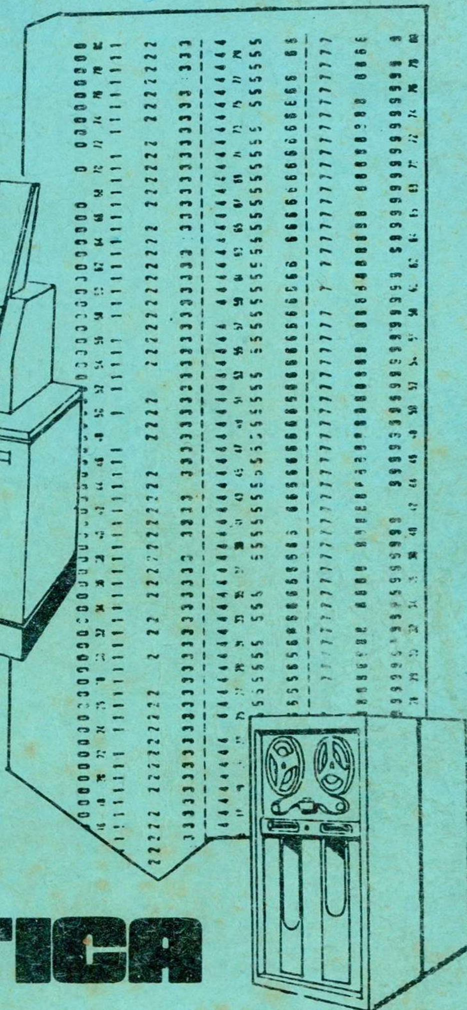
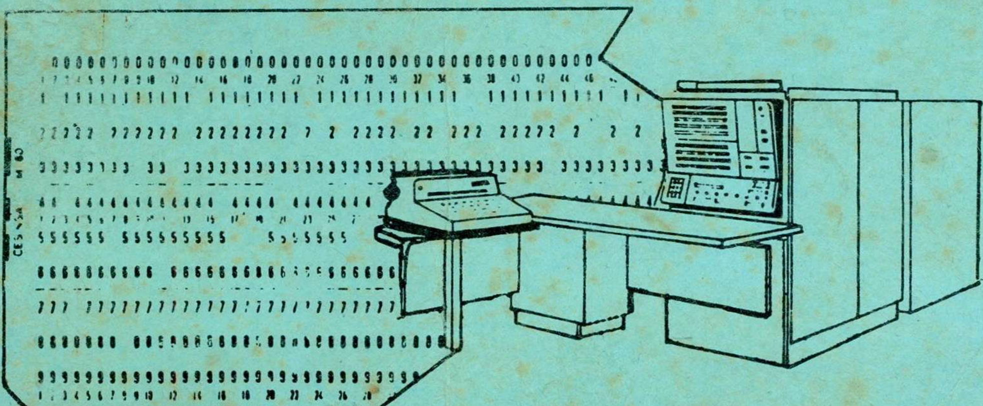


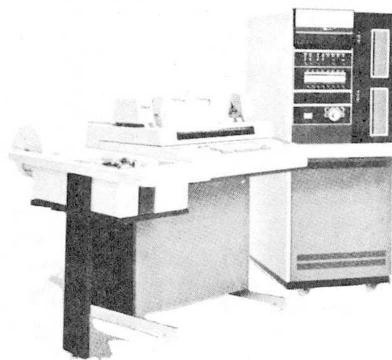


# INICIACION A LA

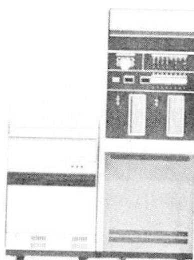
# INFORMATICA



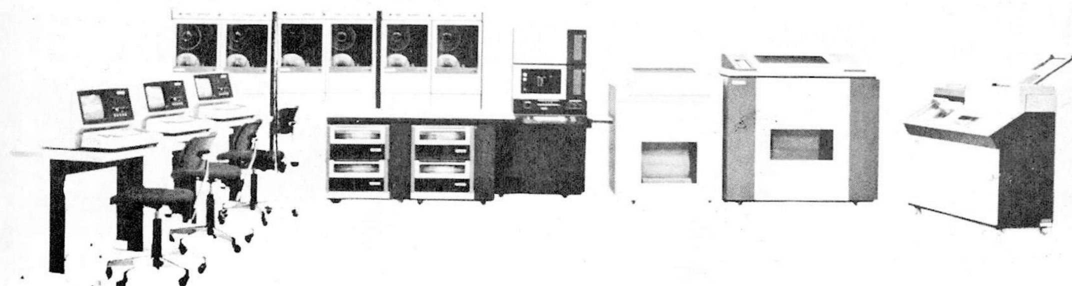
# INTRODUCCION A LA INFORMATICA











## ADVERTENCIA

Algunos textos de los que aparecen en este manual se han tomado, con la debida autorización de las siguientes publicaciones:

- Fronteras de la Ciencia (parte de Informática), publicado por la Universidad de Educación a Distancia y escrito por A. Kubusch y L. Cárdenas.

- Informática Básica, capítulos de Periféricos y Sistemas Operativos, publicados por el Instituto de Informática (Ministerio de Educación y Ciencia) y escritos por A. Kubusch y R. Fuentes.

  Los autores citados son personas que prestan sus servicios actualmente en SECOINSA:

  A. Kubusch: Director de Marketing

  L. Cárdenas: Servicio de Desarrollo de Mercados

  R. Fuentes: Jefe de Educación.

# INDICE

Página

Página

Página

# CAPITULO 1. INTRODUCCION AL PROCESO ELECTRONICO DE DATOS

## 1.1. Ordenadores e Informática

El ordenador es una herramienta capaz de realizar una enorme variedad de trabajos siempre que se le den las "instrucciones" adecuadas en cada caso. El conjunto de instrucciones necesarias para resolver un determinado problema constituye un **programa**, y a las personas que escriben estas instrucciones en un **lenguaje** inteligible por el ordenador se les llama **programadores**.

La ciencia que estudia el tratamiento automático de la información es la Informática.

La historia de la informática y de la industria con ella relacionada comienza apenas hace un cuarto de siglo; a pesar de ello es previsible que en la presente década dicha industria se sitúe, en los países más desarrollados, en tercer o cuarto lugar de acuerdo con su importancia, superada únicamente por las industrias de los sectores siderúrgico, petróleo y automóvil.

La informática ha pasado a ser una técnica de utilización masiva, perdiendo poco a poco, su carácter casi-mítico inicial. Sus implicaciones en una serie de sectores industriales, espacial, militar, siderúrgico, etc., son cada vez mayores; los recursos humanos y materiales que se invierten en este campo son cuantiosos. En cuanto a su futuro todo parece indicar que el mercado se halla aún muy lejos de la saturación, y es continua la aparición de nuevas posibilidades de aplicación.

En lugar de proseguir en un intento de aclaración o de definición de lo que se entiende exactamente por **información**, **tratamiento** y **medios automáticos**, veamos unos cuantos ejemplos típicos del empleo de la informática. Así, un sistema que controla las existencias y los movimientos de un almacén, detec-

tando cuándo deben solicitarse determinadas mercancías, seleccionando el proveedor más adecuado y la cantidad más conveniente, o un sistema capaz de calcular la trayectoria precisa para hacer llegar un vehículo a la luna, o un sistema bancario con la posibilidad de tener registrados los estados de las cuentas de todos los clientes del Banco, actualizando al instante cuantas cuentas queden afectadas por algún movimiento, o un sistema de reserva de billetes de una compañía aérea o de ferrocarril, etc, etc.

En los diferentes ejemplos se trata información de diferentes tipos —numérica, alfanumérica, constante, variable, etc.—, siendo el tratamiento igualmente de muy diferentes formas —manipulaciones aritméticas, comparaciones, clasificaciones, etc.—; la única constante de los ejemplos anteriores es la presencia de un **equipo** muy potente al que se conoce con el nombre de **ordenador electrónico***, capaz de realizar "automáticamente" las anteriores funciones. De ahí que informática y ordenador electrónico aparezcan, invariablemente, estrechamente unidos. Es posible que sin el desarrollo de estos equipos la palabra informática no se hubiese encontrado necesaria y no hubiese surgido, si bien no puede negarse un cierto grado de automaticidad a otros equipos que, en épocas pasadas, fueron desarrollados por el hombre y que, en una escala más o menos modesta, "trataban automáticamente la información" (al menos cierto tipo de ella).

La informática puede ser estudiada desde diferentes puntos de vista: estudio de la herramienta en sí misma, el ordenador, lo que a su vez permite diferenciar enfoques —arquitectura, modo de funcionamiento—, aplicaciones de los ordenadores, modo de resolución de los problemas con ayuda de un ordenador, etc.

En los capítulos 2 y 3 se exponen en detalle los elementos o unidades que componen un ordenador, profundizando más en su función que en su arquitectura fisicoelectrónica. El capítulo 2 trata de la **Unidad Central**, mientras que el capítulo 3 lo hace de la periferia o conjunto de elementos **periféricos**. Es decir, se ciñen a la descripción del llamado **hardware**** del ordenador.

El capítulo 4 ilustra la forma en la que el ordenador almacena internamente los datos para su tratamiento.

Los **lenguajes de programación**, mediante los cuales se proporcionan al ordenador las instrucciones para su trabajo, se estudian en el capítulo 5, mientras en el 6 se exponen las estructuras y organizaciones utilizadas por el tratamiento de grandes cantidades de información.

---

*No es esta la única denominación actualmente empleada. Son también frecuentes las siguientes: **computador (o computadora) electrónico; equipo electrónico de proceso de datos,** etc., habiendo, afortunadamente, caído en desuso el término **cerebro electrónico**, propio de una primera época de mitificación de estos equipos. Además es muy frecuente que se prescinda del adjetivo electrónico y mencionar solamente ordenador o computador.

**Palabra inglesa (**hard** = duro) que viene a significar **todos los componentes físicos** del ordenador.

El capítulo 7 estudia el **Sistema Operativo** de un ordenador, los programas utilizados para mejorar el manejo del equipo y su eficacia, es decir, el **Sofware***.

Los capítulos anteriores constituyen un estudio de "la herramienta". En el capítulo 8 se explica "cómo se usa" dicha herramienta; qué pasos son necesarios para resolver un problema con ayuda de ordenador, permitiendo comprobar que existe un trabajo previo de enorme importancia, sin el cual el ordenador sería prácticamente inútil.

El capítulo 9, por último, estudia los sistemas de **Teletratamiento**: métodos por los cuales se envía información a grandes distancias y sistemas de trabajo con este tipo de equipos.

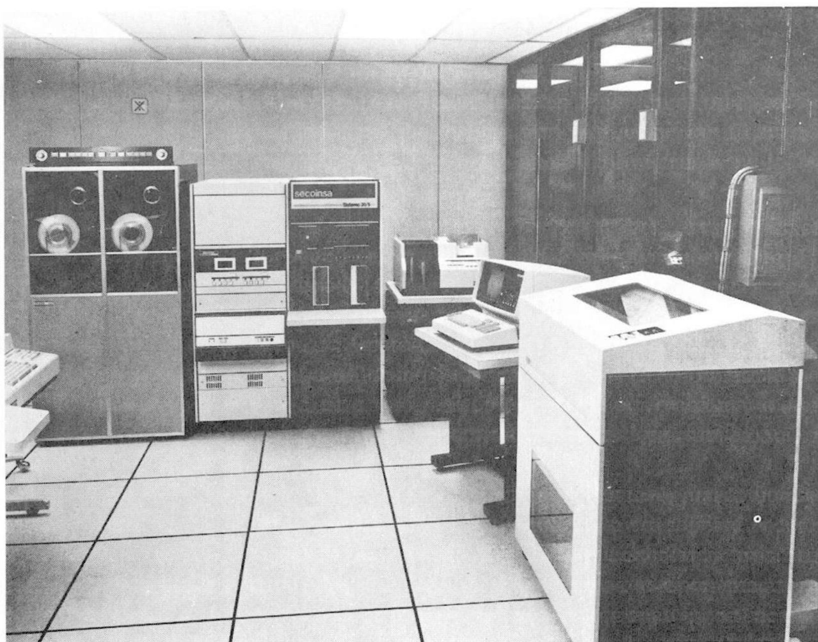



Foto 1.1. Ordenador electrónico

*En contraposición al **hardware**, se crea la palabra **sofware** (en inglés, soft = blando), que expresa toda la parte "no estructural" necesaria para el trabajo de un ordenador.

## 1.2. Mecanización de tareas rutinarias

Durante siglos, y más concretamente durante la pasada época, el ser humano ha invertido gran parte de su actividad intelectual en realizar cálculos de todo tipo, así como largas operaciones de índole administrativo-burocrática y en general trabajos rutinarios que le resultaban indispensables para llevar a término multitud de tareas. El ordenador nace sobre la idea de **liberar** al hombre precisamente de aquello **tedioso** y **largo** para que pueda dedicar mayor parte de su actividad cerebral a tareas más fecundas y creativas.

Por tal motivo, los ordenadores son máquinas diseñadas para poder realizar, en general, **trabajos rutinarios**, estando dotados de gran capacidad para ejecutar a **gran velocidad** un pequeño número de **operaciones elementales** distintas.

Muchos caminos podrían haberse elegido para introducir al lector en el concepto del ordenador, pero parece lo más idóneo presentar como ejemplo típico el de la mecanización de una tarea administrativa.

Para ello se partirá de una oficina de cierta empresa encargada de realizar diversas tareas administrativas (Figura 1), entre ellas y como ejemplo una facturación.

La oficina en cuestión ocupa una habitación. En dicha habitación existe una ventanilla de recepción o **entrada** (1), por donde el empleado (2) recibe diariamente los albaranes que contienen la lista de pedidos efectuados a la empresa.

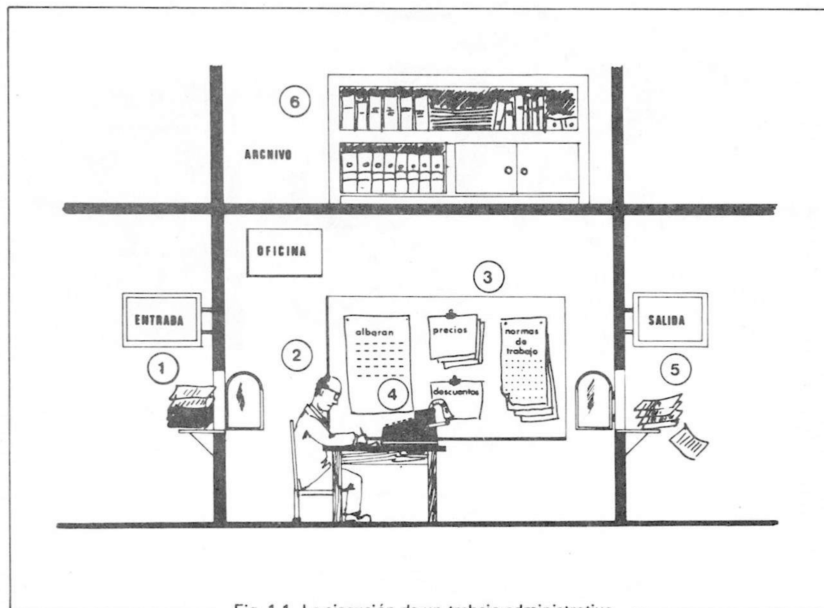



Fig. 1.1. La ejecución de un trabajo administrativo.

El empleado en cuestión, aunque hábil y trabajador, tiene "muy mala memoria", por lo que necesita tener "in situ", y a la vista, todas las normas detalladas para efectuar su trabajo, así como cuantos datos e información sean necesarios para el satisfactorio cumplimiento de su tarea. Para ello se ayuda de un tablero (3), donde coloca en primer lugar el albarán que toma de la ventanilla de entrada. Cada albarán contiene básicamente el nombre del cliente que efectúa el pedido, el tipo de artículos que solicita y la cantidad de los mismos. Previamente, sobre el tablero, ha situado un papel que contiene la lista de precios de los artículos que suministra la empresa y otro que contiene información sobre el tipo de descuentos que cabe aplicar a algunos clientes. Por último, sobre el citado tablero coloca una lista de las normas que debe seguir para efectuar el trabajo concreto de facturación, es decir, un **programa** detallado que contiene todas las **instrucciones** que deberá seguir cuidadosamente y que le indicarán las distintas **operaciones** necesarias para efectuar esta tarea en concreto.

A fin de facilitar la ejecución de operaciones de tipo aritmético, como son por ejemplo sumas y productos de cantidades, el empleado dispone asimismo de una pequeña calculadora (4) que sitúa sobre su mesa de trabajo.

Supuestos todos estos elementos distribuidos en la forma expuesta, puede resumirse el proceso completo de facturación en los siguientes términos:

- El empleado lee cuidadosamente la lista de instrucciones que figuran en el tablero (o programa de normas para efectuar la facturación), ejecutando una detrás de otra las distintas instrucciones. Antes de pasar a la siguiente instrucción, debe efectuar lo que le ordena la última que leyó.
- El empleado, de acuerdo con la primera instrucción, toma un albarán de la ventanilla de entrada y lo coloca sobre el tablero.

Las siguientes instrucciones le obligan a, sucesivamente, consultar el albarán para examinar el artículo o artículos y la cantidad de los mismos que cada cliente solicita. Con estos datos y los demás situados sobre el tablero, las listas de precios de cada artículo que la empresa distribuye, y siempre de acuerdo con las sucesivas instrucciones del programa, efectúa, una vez encontrado en la lista el precio unitario de cada artículo, las multiplicaciones oportunas de precios por cantidad para deducir el precio total o importe del pedido, efectuando algún tipo de descuento al cliente si éste figura como favorecido en la lista al efecto. Estos sencillos cálculos los efectúa con la pequeña calculadora de mesa.

- Rellena el documento de facturación con los datos y resultados correspondientes al cliente y los artículos solicitados y deposita dicho documento (factura) sobre la ventanilla de **salida** (5).

Estas operaciones vuelve a repetirlas para el siguiente albarán, comenzando de nuevo con la primera instrucción del programa, y así cuantas veces haga falta hasta terminar con el último albarán y su correspondiente factura.

De acuerdo con lo anterior, parecerá que el administrativo en cuestión tiene ya resueltos todos sus problemas a la hora de realizar sus trabajos. Sin embargo es posible que pronto se le plantee el problema de la limitación de espacio disponible en el tablero, para almacenar toda la información necesaria para efectuar su trabajo. Esto representa un grave inconveniente cuando:

a) Los datos a utilizar en un determinado trabajo, como el de facturación, resultan excesivamente numerosos (lista de precios de artículos demasiado voluminosa, lista de descuento aplicada a un número muy grande de clientes, etc.).

b) La lista de normas o instrucciones para efectuar el trabajo concreto (por ejemplo programa para realizar facturaciones) es demasiado extensa dada la complejidad o dimensión de la tarea a realizar.

c) Además del trabajo de facturación debe realizar otros trabajos, por ejemplo la nómina de la empresa, la contabilidad, etc; probablemente el tablero no podrá contener simultáneamente todas las normas correspondientes a los diferentes trabajos.

Por tal motivo el administrativo se ha visto obligado a utilizar algo así como una "ampliación" de su tablero, concretamente una biblioteca o archivo situada fuera de su oficina en una habitación separada (6).

En el citado archivo existen debidamente ordenados una serie de **ficheros** que guardan la información utilizable por el administrativo, y que no puede tener permanentemente sobre el tablero por falta de espacio.

Así, nuestro administrativo pondrá sobre el tablero la **información temporal** suficiente para ejecutar una parte del trabajo encomendado. Cuando los **datos** que esté tratando se hayan "agotado" o bien no sean los adecuados en ese momento, solicitará al archivo que le envíe nuevos datos y los situará encima de los anteriormente empleados, con lo que dispondrá de nueva información a tratar.

El proceso es análogo en el caso de que el administrativo haya ejecutado la última instrucción de la lista de normas o **programa del trabajo** (facturación) que está realizando. Si dicha instrucción no es la última del programa, deberá solicitar al archivo una nueva lista de instrucciones. Recibida ésta y colocada encima de la anterior, procederá a ejecutar de nuevo su trabajo.

Asímismo, cuando haya finalizado la ejecución de un trabajo, podrá solicitar del archivo las instrucciones que componen el programa de la siguiente tarea a ejecutar.

Este ciclo de ejecución-petición de información-ejecución se realizará cuantas veces haga falta hasta completar todo su trabajo.

El lector está ahora en condiciones de poder preveer los diferentes órganos o elementos de que consta un ordenador, partiendo del simple hecho de que dicho ordenador tuviera que "mecanizar" o "automatizar" la oficina administrativa que se ha presentado. La figura 2 presenta los órganos básicos de un ordenador. En ella aparecen:

- El elemento E, u **órgano de Entrada** de la información del exterior al ordenador. Sustituirá al elemento (1) o ventanilla de la oficina, y estará constituido por diferentes dispositivos de entrada o lectura de información que pueden ser desde un teclado de caracteres a una lectora de tarjetas perforadas con información debidamente codificada o a un sensor para captura de información no digital.

- El elemento U.C. o **Unidad de Control**, órgano "director" de todas las tareas y el que supervisa a todo el ordenador, que sustituirá al administrativo (2). Incluirá diversos elementos encargados de interpretar las instrucciones del programa a ejecutar, controlando dicha ejecución, así como la secuencia que deben seguir dichas órdenes.

- El elemento M.C. o **Memoria Central** donde se almacenan **datos** y **programas** que son necesarios para la ejecución de las tareas y que vendrá a sustituir al tablero de la oficina (3). Esta memoria tiene una capacidad limitada, insuficiente normalmente para contener todos los datos y programas que deberá usar el ordenador, y está formada por un conjunto de posiciones de memoria que son capaces de almacenar información binaria, a base de ceros y unos.

- El elemento U.A.L. o **Unidad Aritmética y Lógica,** (ALU en terminología inglesa) que efectúa una serie de operaciones elementales y que viene a reemplazar a la pequeña calculadora de mesa (4).

- El elemento S., u **Organo de Salida** de la información que proporciona el ordenador, resultados de su trabajo; sustituirá a la ventanilla (5), y será una impresora, una perforadora de tarjetas, un dispositivo de grabación de cinta magnética, etc.

- El elemento A., o **Archivo** del ordenador, que almacena tanto datos como programas que el ordenador utiliza de una manera frecuente y que no tienen cabida permanente en la memoria central; viene a sustituir al archivo (6) y serán normalmente tambores o discos magnéticos, y en general cualquier dispositivo de almacenamiento de información.

Lo que anteriormente constituía "oficina" propiamente dicha, lo constituye en el ordenador la U.C.P. o Unidad Central de Proceso (normalmente conocida como Unidad Central, simplemente), CPU en terminología inglesa, que incluye los elementos ya reseñados:

– Unidad de Control.
– Unidad Aritmética y Lógica.
– Memoria Central.

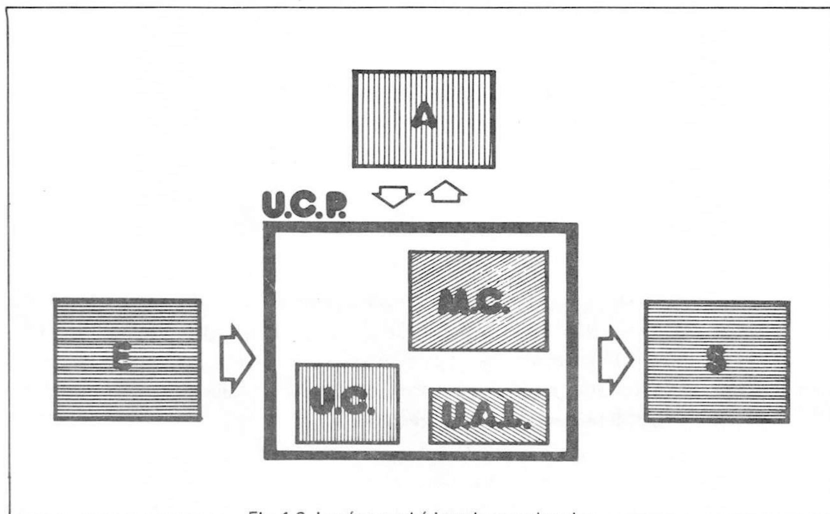



Fig. 1.2. Los órganos básicos de un ordenador.

Todo lo visto constituye la parte material o Hardware del ordenador. Para su funcionamiento, el ordenador deberá poseer además las instrucciones necesarias, el Software. Evidentemente, dado que el ordenador sólo maneja información binaria, a base de ceros y unos, el desarrollo de un programa con este sistema es complejísimo. Para evitar esto se desarrollan los llamados **lenguajes de programación**, con los cuales, mediante instrucciones no binarias, y por ello más manejables por el hombre, se construyen los programas, encargándose otros programas muy complejos, llamados **traductores** de convertir estas instrucciones a aquellas en binario que reconoce el ordenador.

Por otra parte, aún con instrucciones "potentes", una serie de operaciones tales como comenzar un programa cuando termina otro, efectuar traspaso de información de un soporte a otro, etc., resultan ser muy frecuentes y complejas, requiriendo la intervención del operador del equipo para efectuarlas, perdiéndose mucho tiempo en ejecutarlas. Una serie de programas se encargan de controlar

todas estas tareas automáticamente, a la velocidad de trabajo del ordenador. Este conjunto de programas recibe el nombre de Sistema Operativo. De tal forma, el Sistema Operativo, los traductores de lenguajes y los programas dedicados a efectuar un trabajo real, llamados **programas de aplicación**, constituyen el Software del ordenador.

## 1.3. Historia de los instrumentos de cálculo

Si prescindimos de los ábacos, primitivamente usados en diversas civilizaciones, —mediterránea, precolombina, china— o incluso de ingenios mecánicos más modernos —tales como las máquinas de Pascal y Leibniz (siglo XVII)— que podrían considerarse precursoras de las actuales calculadoras de mesa o de bolsillo, en el sentido de, al igual que éstas, no ser "automáticas" en su operación (unas y otras requieren la presencia y acción constante de un operador para introducir datos, provocar las operaciones que deben realizarse con los mismos, apuntar resultados intermedios, etc.), entonces puede considerarse al ingenio ideado por el matemático inglés Charles Babbage, en el siglo XIX, como el verdadero precursor de los actuales ordenadores electrónicos.

Con la máquina ideada por Babbage, y conocida como **Máquina de diferencias**, podrían obtenerse, automáticamente, tablas de valores de una variable Y, en función de otra X, sin necesidad de ordenar operaciones elementales intermedias, tales como sumas, restas, multiplicaciones, etc.

Desafortunadamente la citada máquina, de confección enteramente mecánica, no llegó a construirse nunca por las enormes dificultades que presentaba para su fabricación, a las que la precisión mecánica de aquella época no podía hacer frente.

Por otra parte, Babbage, con una mente desbordante de imaginación, ya había ideado una nueva máquina, la **máquina analítica**, de confección sorprendentemente similar a la de los ordenadores actuales; tal máquina incorporaba los conceptos de Programa, Memoria, Unidad de Control y Unidades de Entrada y Salida. Al igual que en el caso anterior, y por idénticas razones, la máquina no llegó a realizarse; indudablemente Babbage fue un genio demasiado adelantado para la época en que le tocó vivir.

En un sentido absolutamente diferente suelen considerarse las ideas de Herman Hollerith como otro paso fundamental hacia la concepción actual de los ordenadores. Efectivamente, Hollerith, funcionario de la Oficina de Censos de los Estados Unidos, contemplaba el problema de la realización del censo de su país en el año 1890, cuando, a la vista del atraso que se estaba produciendo, era claramente previsible que para ese año aún no se habría acabado de confeccionar el censo correspondiente al año 1880.

Hollerith observó que la mayor parte de las preguntas de que constaban los censos podían contestarse con un Sí o con un No. Ideó entonces la actualmente superconocida tarjeta perforada, consistente en una cartulina en la que, mediante una perforación o ausencia de la misma en unas posiciones predeterminadas, podían plasmarse perfectamente las contestaciones a tal tipo de preguntas. Con el desarrollo paralelo de máquinas cada vez más perfeccionadas a partir de entonces, capaces de detectar y tabular tales perforaciones, la resolución de problemas de un tipo similar a la confección del censo, se aceleró considerablemente.

El nivel necesario en las técnicas auxiliares que, tal como se ha indicado, no encontró Babbage en su día, se presentó ya avanzado el siglo XX. Fue entonces, en 1937, cuando Howard H. Aiken, de la Universidad de Harward (EEUU), siguiendo en gran parte las ideas de su predecesor, fabricó el que realmente puede considerarse primer ordenador, construido utilizando principalmente componentes electromecánicos (ruedas de contador, relés, embragues electromécanicos, etc.). El equipo, conocido como **Automatic Sequence Controlled Calculator** (A.S.C.C., calculador automático de secuencias controladas), y más frecuentemente con el sobrenombre de MARK-I, utilizaba como medio de entrada para los datos tarjetas perforadas. Era una máquina notable por sus dimensiones (17 metros de largo, con un peso cercano a las 70 toneladas) que, aunque sumamente lenta comparada con los equipos actuales, tiene el honor de ser el primer ordenador que llegó a construirse y a funcionar perfectamente.

Siguiendo siempre las ideas primitivas, la incorporación de la electrónica como base para la fabricación de los ordenadores no se hizo esperar. Simultáneamente en diversos países comenzaron a diseñarse y fabricarse ordenadores basados en los componentes electrónicos existentes en aquella época. Los desarrollos que han alcanzado más difusión han sido, sin duda, los de Züse en Alemania y sobre todo los de Eckert y Mauchly en la Universidad de Pensilvania en EEUU. Estos últimos desarrollaron y pusieron a punto en el año 1946 el E.N.I.A.C. (**Electronic Numerical Integrator and Calculator**) que constaba de quince mil válvulas electrónicas de vacío, de tipo similar a las empleadas en los, todavía en uso, equipos de radio y televisión; (parece oportuno recordar que un receptor de radio tiene normalmente alrededor de ocho válvulas). El consumo de este equipo era tal que, en el momento de conectarse, las luces de la ciudad de Filadelfia sufrían un brusco descenso. Su velocidad de trabajo era muy superior al MARK-I (mientras que este último efectuaba una multiplicación de diez cifras en seis segundos, el ENIAC empleaba tres milésimas de segundo).

A partir de entonces los progresos se han venido sucediendo de forma ininterrumpida, las firmas constructoras de ordenadores se han ido multiplicando y los sucesivos modelos de ordenadores han ido apareciendo con gran rapidez, siendo frecuente hablar de **generaciones** de ordenadores; hasta el momento se reconocen tres generaciones, estimándose que la aparición de una cuarta generación está próxima o ya ha sucedido. Las características principales de las tres generaciones pueden ser las siguientes:

**Primera generación.**— 1950-1960. Los ordenadores se construyen con válvulas de vacío, como los iniciales; los constructores miran con mayor atención los campos militar y científico como posibles usuarios de sus equipos. Es una época de enorme mitificación del ordenador, en el que las personas que los manejan se rodean de un cierto aire mítico-misterioso.

**Segunda generación.**— 1960-1965. El transistor (desarrollado varios años antes), sustituye a las válvulas. Se ataca decididamente el campo administrativo y de

gestión de las empresas. Los ordenadores comienzan a ser accesibles a un elevado número de éstas.

**Tercera generación.**— 1965-197?. No aparecen tan claras las fronteras en lo que respecta a los componentes. Se hace uso cada vez mayor de los circuitos integrados, cada vez más densos. La característica principal de los equipos de la tercera generación es la atención dedicada al software, esforzándose los constructores en facilitar a los usuarios de los equipos la programación y explotación de los mismos.

Se construyen ordenadores más pequeños y de menor precio accesibles a la mayor parte de las empresas.

# CAPITULO 2. LA UNIDAD CENTRAL DE PROCESO

## 2.1. Memoria Central

a) Funciones de la Memoria Central

La **memoria** es el elemento del ordenador cuya función es **almacenar datos y programas,** (información en general). La memoria es utilizada en el ordenador para introducir datos del exterior y dejarlos registrados, para albergar el programa que contiene las instrucciones que han de ejecutarse y asimismo para almacenar resultados parciales o intermedios, o bien resultados finales del trabajo que realice. Por lo tanto la memoria podrá considerarse un órgano pasivo de almacenamiento de información en el que se puede:

- Introducir información (''grabar'' o ''escribir en la memoria'').
- Extraer la información (''leer'' de la memoria).

Nótese que toda la información que **procesa** el ordenador tiene que pasar por la memoria, ya sean datos o instrucciones de programa.

b) Estructura de la Memoria Central

La estructura de la memoria se parece mucho a un armario con cajones o a un casillero de correos, puesto que está dispuesto como un conjunto de posiciones o celdas elementales. Cada una de estas celdas puede contener una unidad específica de información, siendo su capacidad variable de unos modelos a otros de ordenadores. En principio basta decir que cada celda podría contener o bien un dato o bien una instrucción de programa. De momento no se insistirá más sobre este punto.

La figura 1 muestra el aspecto simplificado de una memoria. Obsérvese que

cada celda queda identificada por su **dirección** relativa al comienzo de la memoria. Por lo tanto la dirección de cada celda es la que define la situación de cada una de ellas.

Por otra parte, cada celda tiene un **contenido** que es la información que almacena, según se dijo anteriormente.

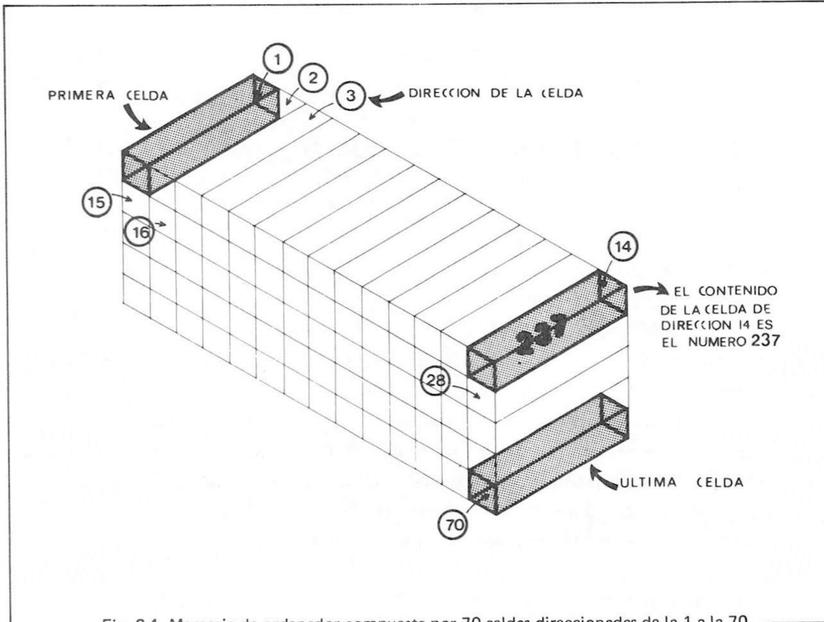



Fig. 2.1. Memoria de ordenador compuesta por 70 celdas direccionadas de la 1 a la 70.

c) Su utilización. Registro de la Memoria Central

Antes de explicar someramente como trabaja la memoria central del ordenador, es necesario hablar de los registros.

Un registro del ordenador es una pequeña memoria unitaria, una celda elemental, que puede recibir una información (datos o instrucciones), conservarla temporalmente, o bien transmitirla a otro lugar del ordenador, según cuales sean las órdenes de la Unidad Central. Son por tanto a modo de soportes temporales de la información que circula por el ordenador.

La figura 2 representa el funcionamiento típico de una memoria, es decir, el proceso de, o bien leer, o bien escribir sobre ella una cierta información.

Para ello la memoria dispone del auxilio de dos registros:

- **Registro de dirección de memoria:** Contiene en un momento dado la dirección de la celda que se trata de buscar o seleccionar entre el conjunto de celdas que componen la memoria.

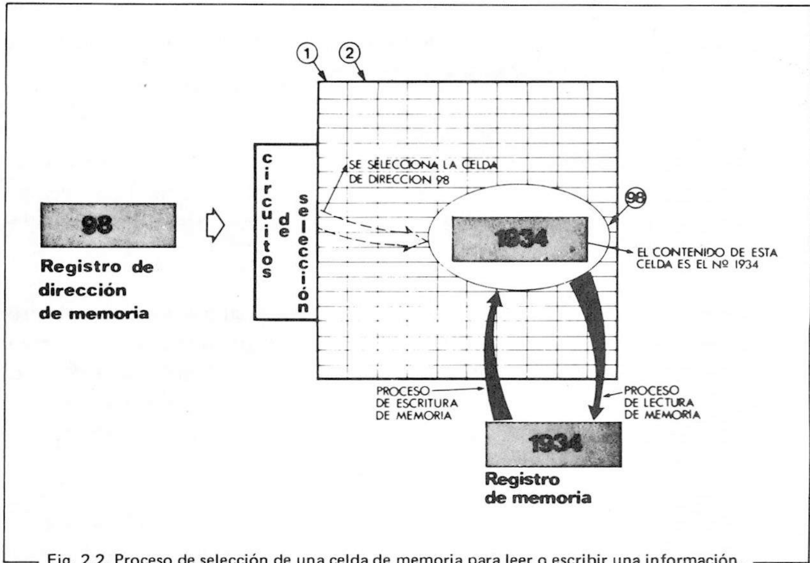



Fig. 2.2. Proceso de selección de una celda de memoria para leer o escribir una información.

- **Registro de información de memoria:** En él deposita la memoria el contenido de la celda seleccionada en el caso de una operación de lectura; o bien contiene una información para que sea depositada en la celda seleccionada en el caso de escritura.

El registro de dirección de memoria proporciona la dirección a que se trata de acceder y unos circuitos especiales se encargan de escoger la celda que tiene esta dirección.

Una propiedad interesante de la Memoria Central es que cuando se "graba" o "escribe" una celda con un dato o instrucción, la información que poseía anteriormente queda sustituida por la introducida. Se puede decir pues que la escritura en la memoria es "destructiva" de la información que anteriormente poseía, tal y como suele ocurrir con un magnetofón.

No ocurre lo mismo, de cara al usuario, con la "lectura" de una celda de memoria; la información contenida en ésta queda inalterada después de su lectura; se puede decir pues que la lectura en memoria, a nivel de usuario, no es "destructiva" de la información. Por este hecho, se puede leer una celda de memoria cuantas veces haga falta a lo largo de un programa sin variar para nada el contenido de la misma.

La figura 2 esquematiza el proceso de lectura y de escritura de memoria con la ayuda de los registros de dirección y de memoria.

d) Constitución física de las memorias centrales

Ya se presentó anteriormente como unidad de información de la memoria, en su aspecto funcional, la celda elemental (que contenía una cierta información). Ahora se trata de ir un poco más lejos e investigar cómo están constituidas las memorias materialmente.

Si el lector pudiera "abrir" un ordenador y contemplar su memoria, lo más probable es que se encontrase con un bloque cúbico constituído por un enjambre de hilos metálicos finísimos entrecruzados. Si pudiera observar más de cerca este bloque descubriría que en las intersecciones de dichos hilos existen unos minúsculos anillos. Dichos anillos son los **núcleos de ferrita**, que son la base de las memorias más usuales.

Dichos núcleos o anillos están constituidos por un material magnético capaz de imanarse en dos sentidos diferentes, según sea el sentido de un impulso de corriente que circule por un hilo conductor que los atraviese (Figura 3); asimismo otro hilo que atraviese los anillos es capaz de examinar (leer) el sentido de imanación que anteriormente adquirieron.

Este proceso de imanación se produce en millonésimas de segundo. Dependiendo del sentido en que circule la corriente eléctrica por el hilo conductor, el núcleo adquirirá una u otra imanación (positiva o estado 1, o negativa o estado 0). Este tipo de trabajo de la memoria se denomina de modalidad binaria (como podría ser el de unas lámparas que puedan estar encendidas o apagadas, unos interruptores abiertos o cerrados).

No solamente las memorias de ferritas trabajan de esta manera, sino todos los elementos que constituyen la "circuitería" del ordenador.

El lenguaje binario resulta idóneo para los ordenadores electrónicos debido a la relativa facilidad con que la tecnología electrónica es capaz de crear dos estados estables (0 y 1), y a la sencillez que supone tanto la representación interna de información como el diseño de circuitos capaces de realizar operaciones en sistema binario.

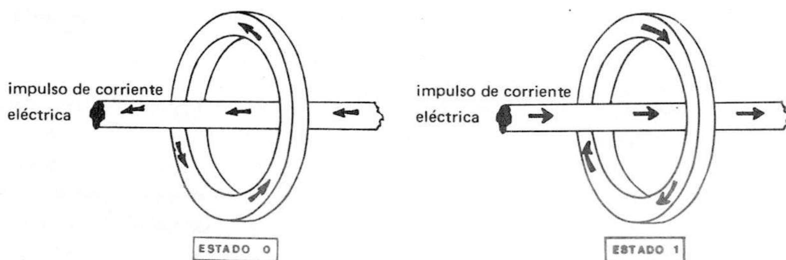



Fig. 2.3. Los núcleos de ferrita se imanan en uno u otro sentido dependiendo del sentido de la corriente que los atraviese, proporcionando dos estados posibles denominados **uno** y **cero**.

Foto 2.1. Conexiones en una placa de memoria

A los dígitos 0 y 1 se les denomina BITS, bit 0 y bit 1 (contracción del inglés "Binary Digits", o dígitos binarios), con lo que el BIT viene a representar la unidad más elemental de información o "átomo" de la información en los ordenadores.

Volviendo ahora a la celda de memoria, no queda sino decir que físicamente está constituida por un conjunto fijo de núcleos de ferritas (variando su número en los distintos modelos de ordenador), por lo que dicha celda contendrá, en general, varios bits de información. Señalemos que existen otros tipos de memorias, como las ya muy frecuentes de **semiconductores**, constituidas por un conjunto de circuitos biestables (también llamados por la literatura técnica anglosajona de "flip-flop": para arriba o para abajo).

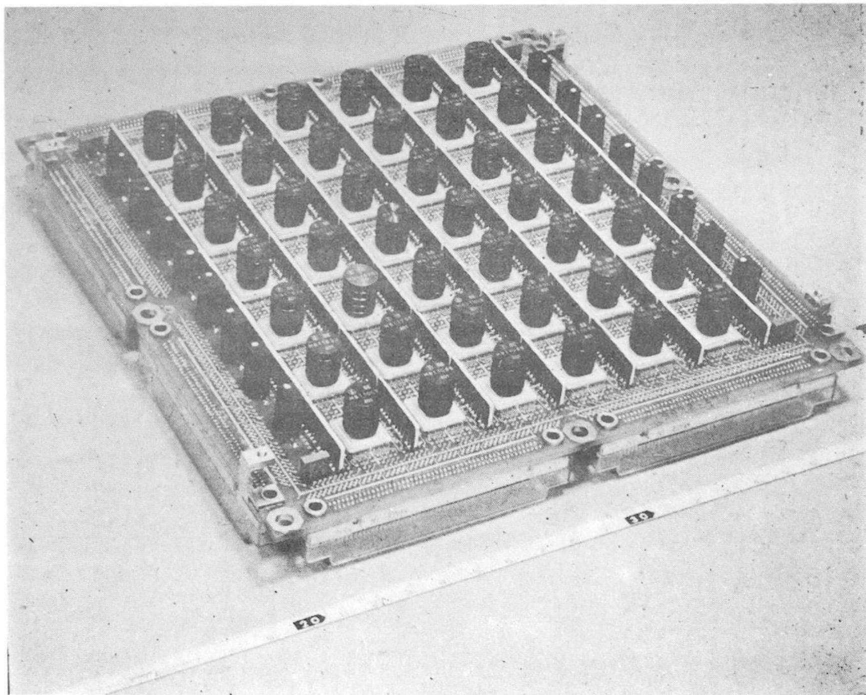


Foto 2.2. Placa de circuitos LSI.

e) Magnitudes más importantes de las memorias centrales

Para terminar este apartado resta hablar de las magnitudes características que definen cualitativa y cuantitativamente a las memorias:

— **Capacidad** o tamaño de las memorias; indica el número de celdas o posiciones que contiene. Normalmente la capacidad de las memorias se mide en miles de posiciones o "K-posiciones de memoria" (por decirlo así kilo-posiciones de memoria). De esta forma decir que un ordenador posee una memoria de 64 K, es decir que tiene 64.000 posiciones de memoria (*).

La capacidad de un ordenador puede variar entre amplios límites,existiendo desde ordenadores muy pequeños con una capacidad de solamente 4 K hasta ordenadores muy grandes con capacidad de varios miles de K. (A veces la capacidad de memoria se utiliza como parámetro calificador del tamaño de un ordenador; así los ordenadores pequeños pueden tener entre 4 y 32 K, los de tipo medio entre 32 y 128 K y los ordenadores grandes, capacidades superiores).

— **Tiempo de acceso.** Es el tiempo que invierte el ordenador en acceder a una posición (celda) de memoria para leer una información, o bien el tiempo que tardaría en escribir una información sobre una posición de memoria previamente seleccionada. Tan cortos son los tiempos que invierte el ordenador en realizar estas operaciones, que dichos tiempos se suelen medir en:

microsegundos ($\mu$s) =

$$= \frac{1}{1.000.000} \text{ seg.} = 10^{-6} \text{ seg.}$$

o incluso en

nanosegundos (ns) =

$$= \frac{1}{1.000.000.000} \text{ seg.} = 10^{-9} \text{ seg.}$$

Los tiempos de acceso de los ordenadores actuales pueden oscilar (según modelo) entre 0,2 $\mu$s y 2$\mu$s (como ejemplo significativo, un ordenador con una memoria de 100.000 posiciones y un tiempo de acceso de 500 ns podría leer o escribir la memoria entera en media décima de segundo).

— **Tamaño de la celda o posición.** Respecto a la dimensión de las celdas de memoria, se tienen entre las más corrientes los tipos siguientes:

(*) En realidad una K no son exactamente mil posiciones, sino $2^{10} = 1024$, pero se suele tomar como 1000 por comodidad.

– **Celdas de 6 bits.** Los signos o caracteres que necesita poder almacenar y tratar un ordenador, son por decirlo así, los que existen en una máquina de escribir (letras A, B, C,. . . . . Z, dígitos decimales 0,1. . . . . 9 y otros caracteres como ?;/ ( ) etc.), es decir, los necesarios para tratar información escrita. En las celdas de 6 bits se emplean códigos binarios en que se asigna a cada una de las 64 combinaciones posibles que pueden formarse con los 6 bits, un determinado carácter, número de una cifra o letra. Por ejemplo:

000000 → A

000001 → B

100111 → 1

101101 → 2

110000 → 9

111100 → ?

111011 → ; etc.

Existen diversos códigos de 6 bits que asignan combinaciones diferentes a la colección de signos existentes; todos ellos pueden representar todos los signos, pudiendo incluso sobrar combinaciones de las 64 existentes, sin asignación de carácter alguno.

– **Celdas de 8 bits** (también llamadas "octeto" y "byte"). Todo lo explicado anteriormente para la celda de 6 bits, es válido para la de 8, pero lógicamente existirán más combinaciones posibles, en concreto desde

0000 0000 → 0 (decimal)

0000 0001 → 1 (decimal)

. . . . . . . . . . . . . . . hasta

1111 1111 → 255 (decimal)

es decir, 256 combinaciones diferentes. Naturalmente que se podrán representar colecciones de caracteres más numerosos, utilizando diferentes códigos: ASCII, EBCDIC, etc.

Existe además la posibilidad de almacenar **dos cifras** (del 0 al 9) **empaquetadas** en un octeto. Cada octeto se divide en dos "cuartetos" (unidades de 4 bits), cada uno de los cuales podrá representar una cifra decimal.

0000 → 0
0001 → 1
................. hasta
1001 → 9

con lo que en el octeto se podrán almacenar cifras de la siguiente forma:

0000 0000 → 0 0

0000 0001 → 0 1

1001 1001 → 9 9

– **Celdas de 12, 16, 24, 32 y 36 bits.**
Así como los ordenadores que utilizan celdas de 6 y 8 bits se denominan "orientados a caracteres", los que emplean celdas mayores, como las expuestas, se denominan "orientados a palabras" (*)

Como ejemplo, una celda de 32 bits puede almacenar:

– 4 bytes u octetos (4 caracteres)
– Un número binario entre 0 y $2^{32}-1$
– 8 cifras decimales "empaquetadas".

(*) En general, un ordenador orientado a caracteres necesitará de 4 posiciones de memoria para almacenar un número de 4 cifras o un nombre de 4 letras, mientras que un ordenador orientado a palabras, podrá almacenar en una sola posición cantidades muy grandes (de varias cifras decimales) o nombres de varias letras.

## 2.2. Unidad Aritmético-Lógica

La U.A.L. es el órgano del ordenador encargado de efectuar operaciones:

- Aritméticas: (suma, resta, multiplicación, división).
- Lógicas: (comparar, tomar decisiones ante situaciones determinadas del programa, etc.).

Todas estas operaciones se realizan con datos o información que previamente ha estado almacenada en la memoria. La U.A.L. está compuesta de ciertos elementos o circuitos que se encargan de efectuar operaciones muy sencillas. Nunca maneja más de **dos** datos cuando efectúa una operación aritmética; así, en el caso de tener que sumarse tres cantidades, necesita realizar la operación en dos pasos sucesivos.

La U.A.L. efectúa todas las operaciones a base de unos circuitos electrónicos que, debidamente conectados (previa orden de la unidad de control), efectúan la operación encomendada. En cualquier caso se trata de operaciones sencillas, como suma y resta de cantidades. Como ejemplo, muchos ordenadores no disponen de circuitos que realicen la multiplicación y división (por lo tanto, ante la ausencia de estos, habrá que realizarlas a base de sumas y restas; para una persona esto resultaría sumamente tedioso, pero no para el ordenador que trabaja a velocidades de cálculo de cientos de miles de instrucciones por segundo).

A fin de centrar las ideas sobre los elementos que constituyen una U.A.L., se remite a la Figura 4. En ella se presentan los elementos básicos que requeriría una U.A.L. para efectuar operaciones aritméticas con dos cantidades. Estas cantidades serían pues, los operandos de cierta operación aritmética. De momento no se va a profundizar sobre cómo estas dos cantidades han ido a parar a la U.A.L.

Unicamente se debe observar que, como soporte de dichas cantidades, aparecen dos registros (similares a los que se presentaron al hablar de la Memoria Central):

R — OP1: Registro que recoge el operando 1 (en el caso de una suma sería el primer sumando).

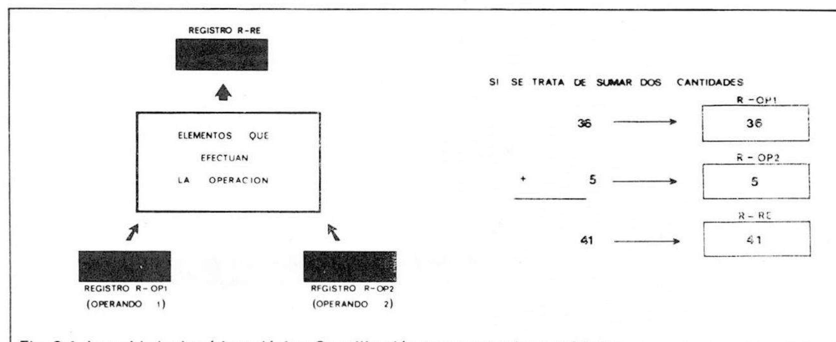


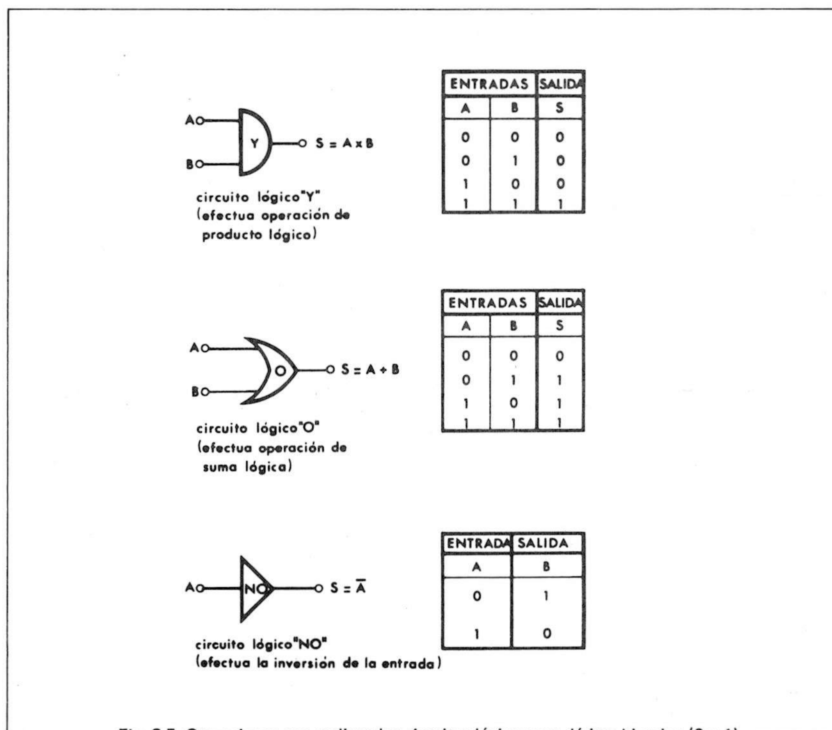

Fig. 2.4. La unidad aritmética y lógica. Su utilización para sumar dos cantidades.

R – OP2: Registro que recoge el operando 2 (en el caso de una suma sería el segundo sumando).

R – RE: Registro que recoge el resultado de la operación.

Entre ellos se sitúan los elementos o circuitos que realizan esta operación aritmética. Estos circuitos están especialmente diseñados para tratar cantidades en forma binaria (unos y ceros, únicos que trata el ordenador) y están basados en los llamados circuitos lógicos (Figura 5).

La figura 7 muestra detalladamente cómo lleva la información cada registro, en el caso de una suma de dos cantidades. (Obsérvese que las cantidades binarias representadas en la figura 6 son las mismas que las indicadas en decimal en la figura 4).




| ENTRADAS | | SALIDA |
|---|---|---|
| A | B | S |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| ENTRADAS | | SALIDA |
|---|---|---|
| A | B | S |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| ENTRADA | SALIDA |
|---|---|
| A | B |
| 0 | 1 |
| 1 | 0 |

Fig. 2.5. Operaciones que realizan los circuitos lógicos con dígitos binarios (0 y 1).

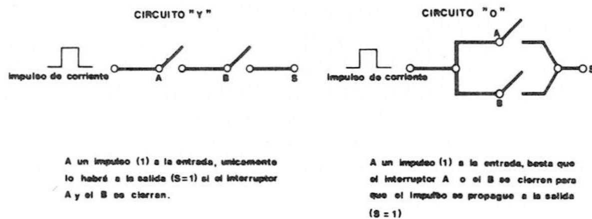



Fig. 2.6. Ejemplo de la realización física de los circuitos lógicos mediante interruptores.

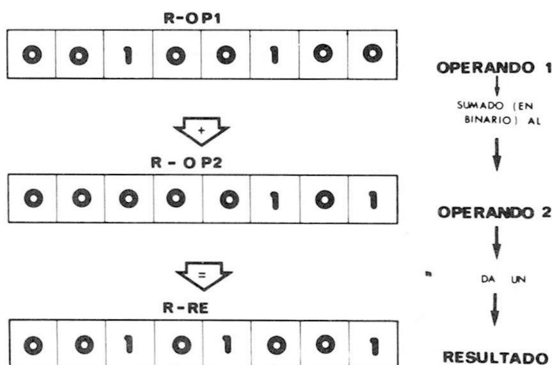



Fig. 2.7. Los registros de la U.A.L. son pequeñas memorias de un reducido número de bits (8 en este ejemplo) Sirven de almacenamiento temporal de cantidades binarias con las que efectúan operaciones.

## 2.3. Unidad de Control

### 2.3.1. Estructura de la Unidad de Control

Anteriormente se dejó constancia de que la U.C. era el órgano "más inteligente" del ordenador, que dirigía todas las operaciones del mismo. Se apuntó asimismo, que esta acción la ejercía en base a las instrucciones que contenía el programa almacenado en la Memoria Central. Es decir, la Unidad de Control:

- Analiza e interpreta las instrucciones del programa que se está ejecutando.
- Dirige el funcionamiento de las restantes unidades del ordenador mediante órdenes dirigidas a las mismas, cuando lo considera oportuno (controla pues, los tiempos de actuación del resto de las unidades).

La figura 8 muestra un esquema simplificado de una Unidad de Control.

Las partes más importantes de aquélla son las siguientes:

– **Reloj.**
Para la correcta distribución en el tiempo de cada operación que se ejecuta en el ordenador y medir los momentos en que cada una debe comenzar y aquéllos en que debe terminar, la Unidad de Control dispone de una especie de reloj o cronómetro que proporciona una sucesión de **pulsos** a intervalos fijos de tiempo que sirven como referencia al ordenador.

– **Registro Contador de Instrucciones.**
También llamado Contador de programa, contiene la dirección de memoria donde se encuentra la **siguiente** instrucción del programa almacenado que se va a ejecutar.

El valor de este contador irá aumentando una cantidad cada vez que una instrucción es ejecutada (esta cantidad será la unidad en el caso de que cada instrucción del programa ocupase una celda de memoria).

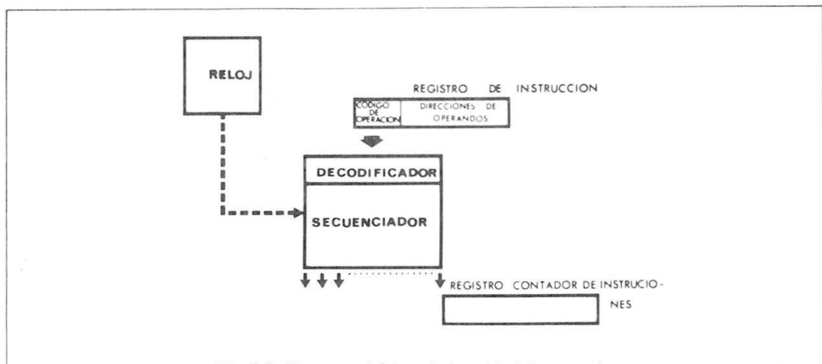


Fig. 2.8. Elementos básicos de la unidad de control.

Excepción a esta norma se da cuando se produce la ejecución de una instrucción de salto o bifurcación, que situará en el registro contador de instrucciones la dirección de memoria a la que se pretende acceder con dicho salto o bifurcación.

— **Registro de instrucción.**
En él se deposita la instrucción del programa que se está ejecutando. La unidad de control no hace sino ejecutar instrucción tras instrucción del programa. Para ello, cuando le llega el turno a una de ellas, procede a su traslado desde la memoria a este registro.

— **Decodificador y secuenciador.**
El decodificador es un elemento encargado de **analizar** el código de operación de la instrucción depositada en el registro de instrucción. Analiza el código de operación para determinar cual es la operación a efectuar y además investiga la parte de dirección de operandos para determinar con qué datos hay que efectuar las operaciones (con objeto de ir a buscarlos a la memoria).

En base a estos análisis, el secuenciador se encarga de generar unas órdenes muy simples llamadas microórdenes, sincronizadas con los pulsos del reloj y las distribuye a los diversos elementos o unidades para que, paso a paso, se ejecute la instrucción cargada en el registro de instrucción.

Se distinguen dos tipos de secuenciadores:

— Secuenciadores cableados que están diseñados a base de circuitos electrónicos fijos que envían siempre las mismas microórdenes para cada instrucción que se ejecute.
— Secuenciadores microprogramados. En este caso el secuenciador dispone de una pequeña memoria que contiene un **microprograma** para cada instrucción del repertorio. De manera que se trata de un secuenciador "microprogramable" con lo que en determinados casos, podrán variarse los microprogramas generándose microórdenes distintas; los ordenadores que incluyen secuenciador de este tipo se denominan ordenadores de lógica programada o simplemente ordenadores "microprogramables" (Firmware)

### 2.3.2. Instrucciones. Tipos

El lector conoce ya lo que es básicamente una instrucción de programa. Todas ellas contienen dos partes fundamentales:

- **Código de operación** que indica a la U. de Control **lo que debe hacer** (por ejemplo sumar, acudir a una dirección de memoria, parar el programa, leer una ficha perforada de una Unidad de entrada, etc.)

- **Dirección de operando** (u operandos), indica **dónde están situados los datos** que debe manejar para ejecutar la instrucción (normalmente suelen ser una o varias direcciones de memoria).

Las instrucciones operan con un máximo de dos datos u operandos. Una estructura como ésta, permite a una misma instrucción, por ejemplo de sumar, realizar la adición de cualquier pareja de números que se hayan situado previamente en las direcciones de memoria señaladas por la instrucción.

En cuanto al número de instrucciones distintas de que dispone un ordenador, varía de unos modelos a otros. (A esta colección de instrucciones propias del ordenador se les suele denominar **juego o repertorio de instrucciones**). Los ordenadores muy sencillos pueden tener poco más de una veintena de ellas, mientras que algunos ordenadores especiales poseen cerca de doscientas; en general la mayoría no llegan al centenar. En realidad hacen falta tantas instrucciones como operaciones distintas se deseen efectuar mediante una sola de ellas. Desde luego cuanto más extenso sea el repertorio de instrucciones del ordenador, más confortablemente podrá ejecutar los programas.

Profundizando en las estructuras de las instrucciones se puede decir que, en general, los ordenadores utilizan alguno de los siguientes tipos:

- **Instrucciones de tres direcciones.**

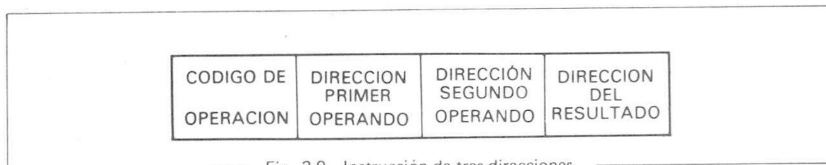


Fig. 2.9. Instrucción de tres direcciones.

En esta estructura, las dos primeras direcciones señalan dónde están situados los dos datos en memoria. La otra dirección indica la dirección donde debe almacenarse el resultado de la operación, una vez efectuada por el ordenador. Los ordenadores que utilizan un repertorio de instrucciones con esta estructura, se dice que tienen una **lógica de tres direcciones.**

—**Instrucciones de dos direcciones.**

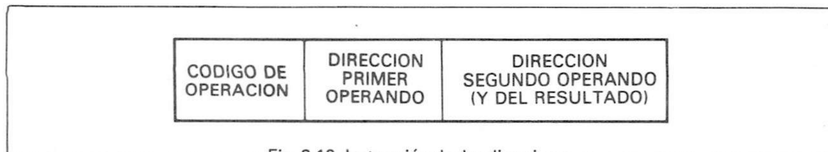



Fig. 2.10. Instrucción de dos direcciones.

Es una simplificación de la de tres direcciones. En la segunda dirección (dirección del segundo operando), se coloca siempre el resultado de la operación; lógicamente, la cantidad o dato que antes de ejecutarse la instrucción constituía el segundo operando quedará destruido después de la operación, quedando sustituido por el dato resultado de la operación. Si se desea impedir esto, es decir, conservar el segundo operando, deberá previamente guardarse éste ("copiarse") en otra posición de memoria mediante una instrucción adecuada.

—**Instrucciones de una dirección.**

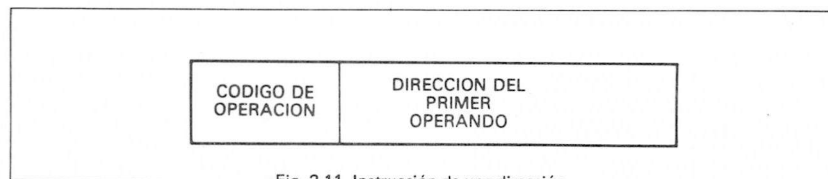



Fig. 2.11. Instrucción de una dirección.

En las instrucciones que obedecen a esta estructura solamente se proporciona una dirección de operando. En algún sitio o de alguna manera, se deberán especificar, por tanto, el segundo operando y la dirección donde guardar el resultado.

Para el funcionamiento de los ordenadores que utilizan instrucciones con esta estructura se precisa la existencia de un **acumulador** (un registro especial de la U.A.L.) que deberá contener, antes de ejecutarse la instrucción, el segundo operando, y que guarda asimismo el resultado de la operación una vez efectuada. Lógicamente, para realizar una operación con dos cantidades primero hay que utilizar otra instrucción que "cargue" el acumulador con uno de los operandos; después de ejecutada la instrucción, habrá que utilizar otra que "descargue" el contenido del acumulador en la dirección de memoria que se desee guarde el resultado de la operación.

El lector comprenderá que cuanto más simple sea la estructura de las instrucciones, más laboriosa resulta la confección de programas por el usuario.

Una clasificación de las instrucciones en su aspecto funcional, obedece, lógicamente, al tipo de código de operación que cada instrucción lleve. Por lo tanto se pueden clasificar en:

– **Aritméticas.** Son capaces de efectuar operaciones entre operandos de suma, resta y algunas veces multiplicación y división.

– **De traslación.** Efectúan movimientos de datos, por ejemplo llevar una cantidad desde el acumulador a la memoria (escritura en memoria), o viceversa, llevar un dato de la memoria al acumulador, o bien mueven datos de unas direcciones a otras, dentro de la memoria.

– **De bifurcación.** O instrucciones que interrumpen el proceso de secuencia de programa.

Las instrucciones del programa se van ejecutando en el orden en que están colocadas en memoria, es decir, una detrás de la otra (proceso de automatismo de ejecución). A llegar a una instrucción de bifurcación, el programa rompe la secuencia y acude a otra instrucción colocada en alguna posición de la memoria señalada, precisamente, por la parte de dirección de la instrucción de bifurcación. A su vez esas instrucciones pueden ser:

– **Condicionales.** Se efectúa una bifurcación a una instrucción almacenada en algún lugar de la memoria si se cumple alguna condición (por ejemplo la última operación efectuada en la Unidad Central dio un resultado negativo).

– **Incondicionales.** Se efectúa una bifurcación sin dependencia de los hechos ocurridos anteriormente a la ejecución de la instrucción.

– **De interrupción o parada.** Existen instrucciones que al ser ejecutadas obligan a parar el proceso de ejecución del programa (por ejemplo una instrucción de final de programa, una instrucción de espera, etc.).

– **De entrada o salida.** Se encargan del trasiego de datos desde o hacia los órganos de entrada o salida, por ejemplo leer una información en una cinta magnética, escribir cierta información sobre una impresora, etc.

### 2.3.3. Proceso de ejecución de las instrucciones.

Supóngase que un ordenador está ejecutando un programa almacenado en su memoria. Si se pudiese parar el ordenador justo en el momento de comenzar la ejecución de una de las instrucciones del programa, y se pudiese seguir a "cámara lenta" el desarrollo de la nueva instrucción, se podrían encontrar claramente diferenciadas dos fases de ejecución. (Las cantidades que figuran como direcciones y contenido de registros en las figuras de este apartado se expresarán en decimal a fin de facilitar al lector la consulta de las mismas, pero no se debe olvidar que, internamente, se trata de cantidades binarias).

Supóngase que la instrucción que empieza a ejecutarse está situada en la posición (o celda) 314 de la memoria. Las dos fases de que consta la ejecución según se dijo anteriormente, son las siguientes:

a) Búsqueda e interpretación de la instrucción que se encuentra en la memoria.
b) Ejecución propiamente dicha de la instrucción.

A su vez cada fase se compone de una serie de pasos elementales.

**a) Fase de búsqueda e interpretación de la instrucción.**

La secuencia de pasos correspondientes a la fase de búsqueda e interpretación de una instrucción, viene ilustrada en la figura 12. Esta secuencia es idéntica para todas las instrucciones del repertorio que puedan estar en el programa.

Los pasos efectuados son los siguientes:

(1) La Unidad de Control envía una microorden para transferir el contenido del registro contador de instrucciones (dirección de la próxima instrucción a ejecutar) al registro de dirección de memoria.

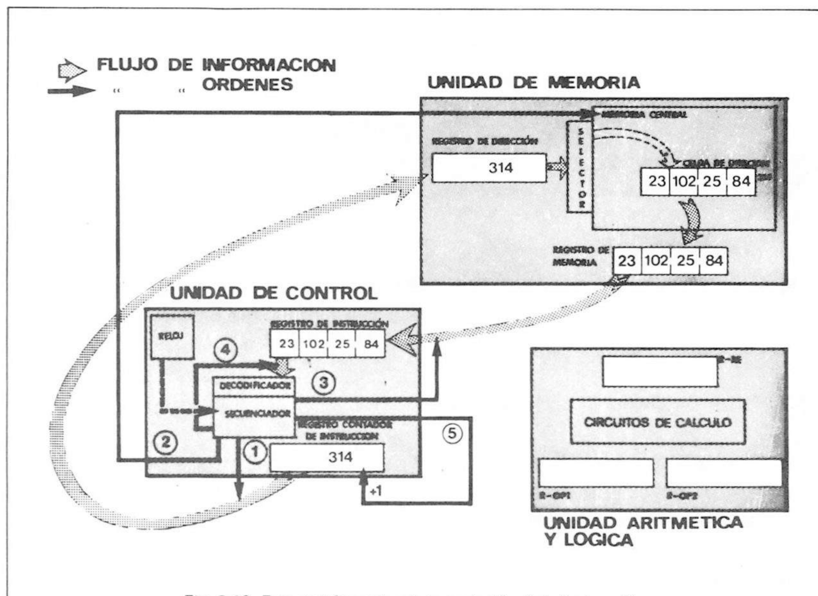



Fig. 2.12. Fase de búsqueda e interpretación de la instrucción.

(2) Se selecciona la posición de memoria cuya dirección contiene el registro de dirección, depositándose en el registro de memoria el contenido de la celda seleccionada, (lectura de memoria). Este contenido, será lógicamente, la instrucción a ejecutar.
(3) Se transfiere el contenido del registro de memoria al registro de instrucción.
(4) A continuación el decodificador procede a la interpretación de la instrucción depositada en el registro de instrucción (parte del código de operación).
(5) El registro contador de instrucciones es incrementado en una unidad, con lo que su contenido al finalizar ésta fase será 315, es decir la dirección de memoria de la siguiente instrucción a ejecutar una vez que haya acabado la ejecución de la instrucción actual.

**b) Fase de ejecución de la instrucción.**

Esta fase comienza recogiendo los operandos a sumar de la memoria, y efectuando dicha suma en la U.A.L. Posteriormente se depositará el resultado en la memoria. Sus fases son:

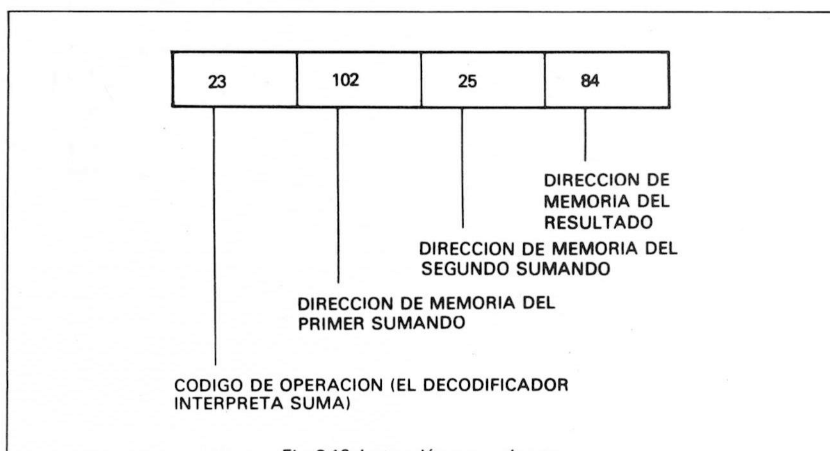



Fig. 2.13. Instrucción que se ejecuta.

(1) La dirección del primer operando es transferida desde el registro de instrucción al registro de dirección de memoria.
(2) Se selecciona la posición de memoria cuya dirección contiene el registro de dirección y se deposita en el registro de memoria el contenido de la celda seleccionada, es decir, el primer operando (lectura de memoria).
(3) Se transfiere el contenido del registro de memoria al registro R-OP1 de la U.A.L.
(4) Se transfiere la dirección del segundo operando desde el registro de instrucción al registro de dirección de memoria.
(5) Se selecciona la posición de memoria que contiene el registro de dirección y se deposita en el registro de memoria el contenido de la celda seleccionada (lectura de memoria).

(6) Se transfiere el contenido del registro de memoria al registro R-OP2 de la U.A.L.

Estos primeros pasos aparecen reflejados en la figura 14; los siguientes, en la figura 15, son:

(7) El secuenciador envía una microorden a la U.A.L. para que efectúe la operación de suma con los dos registros R-OP1 y R-OP2 y almacene el resultado en el R-RE.

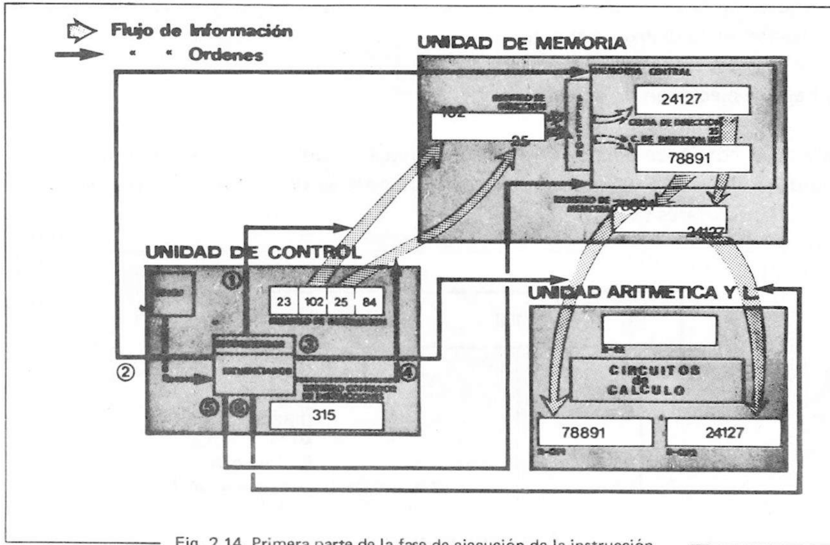



Fig. 2.14. Primera parte de la fase de ejecución de la instrucción.

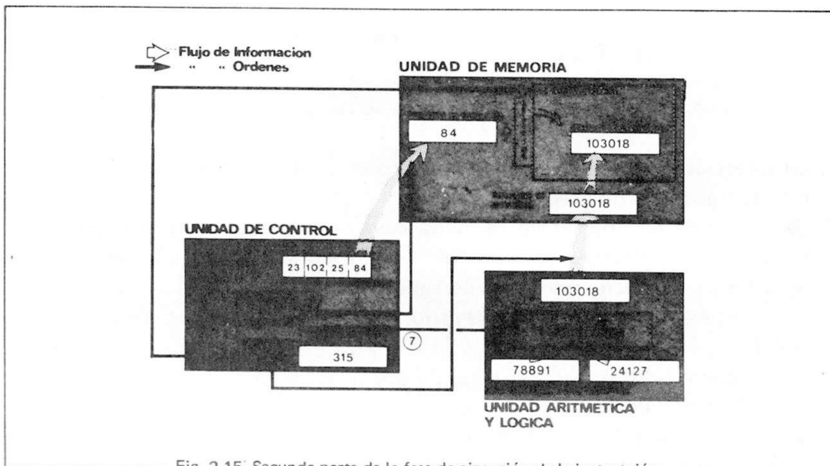



Fig. 2.15. Segunda parte de la fase de ejecución de la instrucción.

(8) El resultado es enviado desde el registro R-RE al registro de memoria.
(9) Se transfiere la dirección donde debe almacenarse el resultado desde el registro de instrucción al registro de dirección de memoria.
(10) Se selecciona la posición de memoria que corresponde al contenido del registro de dirección, transfiriendo a continuación el contenido del registro de memoria sobre la celda seleccionada (escritura de memoria).

Por supuesto los pasos correspondientes a la fase de ejecución de una instrucción son diferentes para cada tipo de instrucción; sería pues necesario presentar todas y cada una de las instrucciones para ver todos los pasos posibles de esta fase. No obstante, para el objeto del presente manual, la presentación de la instrucción anterior de suma se estima como suficientemente representativa para el lector.

El contenido del registro de instrucción es la instrucción que viene representada en la figura 13. Se supone que el código de operación 23 representa la operación suma de dos cantidades (se ha elegido para el ejemplo un ordenador que tuviera lógica de tres direcciones para que el lector pueda seguir más detalladamente el transcurso de la sucesión de pasos).

### 2.3.4. Ejemplo de programa.

A continuación se va a presentar la ejecución de un programa completo. Para ello se van a especificar previamente ciertos elementos que definan, por decirlo así, el ordenador específico que se va a utilizar en el ejemplo.

**a) Estructura de instrucciones y memoria elegida.**

Se partirá de una memoria central de 64 K posiciones (64 x 1.024 = 65.536 posiciones). Estas celdas se supondrán de un tamaño de 24 bits (figura 16).

Cada celda de memoria podrá contener en general:

—Una instrucción del ordenador de 24 bits (tamaño fijo para todas ellas).;
—Un dato también de 24 bits.

**Las instrucciones**

El repertorio de instrucciones del ordenador está compuesto por instrucciones que obedecen a la estructura representada en la parte central de la figura 16.

Se puede observar que se trata de instrucciones de una dirección, por lo tanto el ordenador dispondrá de un acumulador. Se ha preferido presentar el ejemplo con este tipo de instrucción por la sencillez de su manejo y para que el lector tenga la

posibilidad de comparar su funcionamiento con el de tres direcciones que presentábamos anteriormente. En aquel caso se presentaban instrucciones de tres direcciones que resultaban más adecuadas para explicar el proceso de operaciones aritméticas.

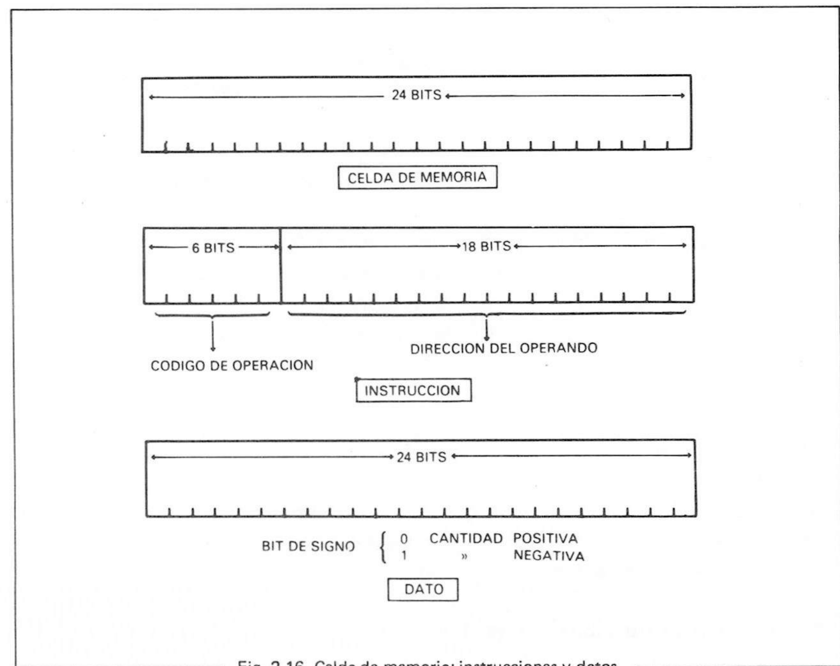



Fig. 2.16. Celda de memoria; instrucciones y datos.

Obsérvese que:

– El **código de operación** es de 6 bits. Con un código de este tipo podremos tener los siguientes códigos de operación distintos:

000000
000001
000010
. . . . . . hasta
111111

En total $2^6 = 64$ combinaciones posibles, es decir,este es el número de códigos de operaciones que podrían darse (correspondiente a otras tantas instrucciones diferentes, que formarán el repertorio).

– **La dirección del operando** de 18 bits daría acceso a las siguientes posiciones de memoria:

000000000000000000
000000000000000001
000000000000000010
. . . . . . . . . . . . . . . . . . hasta
111111111111111111

La última dirección (todos unos) representa en decimal la cantidad 262.143, o última posición de memoria a la que se podría acceder. En definitiva se podrían direccionar celdas desde la 0 hasta la 262.143, es decir un total de 262.144 celdas (256 x 1.024 = 256 K – celdas).

(la memoria presentada para el ejemplo de 64 K– celdas, es pues inferior a la máxima posible direccionable).

**Los datos**

Los datos que pueden utilizarse serán **numericos** exclusivamente y su estructura con arreglo a lo indicado en la figura 16 (parte inferior).

El bit de la izquierda sirve para indicar si la cantidad expresada en los 23 bits restantes es positiva o negativa.

El máximo valor del dato que puede contener una celda, será:

011111 . . . . . . . . . . . . . . 1 = + 8.388.607 (expresado en decimal)
23 bits 1

y el mismo valor negativo

11111 . . . . . . . . . . . . . . 1 = – 8.388.607 (expresado en decimal)

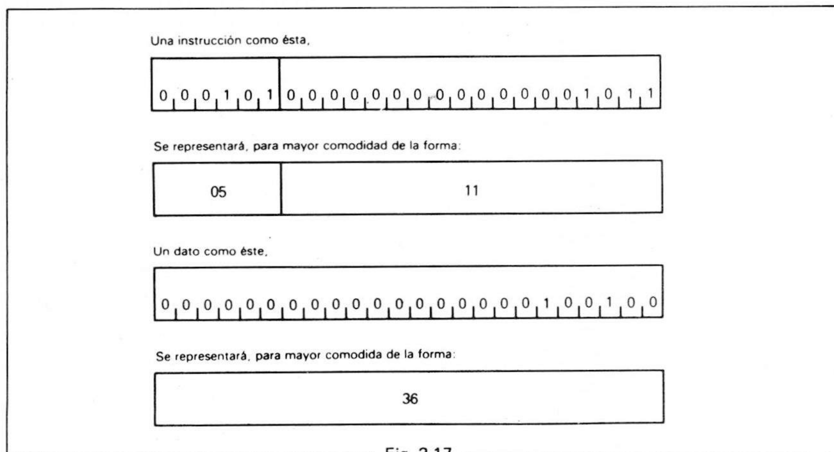


Fig. 2.17.

**b) El juego de instrucciones elegido**

A continuación se presenta una serie de instrucciones, elegidas entre el repertorio de todas aquellas que podría utilizar el ordenador (recordemos que habrá 64 como máximo).

Los números situados a la izquierda de cada instrucción, representan los códigos de operación que la U. Control decodificará para efectuar las operaciones que indican. En la parte derecha de las instrucciones se representa una dirección cualquiera a modo de ejemplo para facilitar su explicación.

Conviene recordar que las instrucciones de una dirección implican la utilización de un registro llamado acumulador, según se explicó al hablar de la Unidad de Control y las instrucciones.

Repertorio de instrucciones

| Instrucción | | Significado |
|---|---|---|
| 05 | 20.314 | **Cargar el acumulador** con el contenido de la celda de memoria cuya dirección es la 20.314. El contenido anterior del acumulador queda destruido. |
| 11 | 103 | **Almacenar en la celda** de memoria cuya dirección es la 103 el contenido del acumulador. (No se altera el contenido del acumulador). |
| 14 | 54.703 | **Sumar al contenido del acumulador** el contenido de la celda de memoria cuya dirección es la 54.703. |
| 22 | 8.000 | **Restar al contenido del acumulador** el contenido de la celda de memoria cuya dirección es la 8.000. |
| 25 | 10.725 | **Multiplicar el contenido del acumulador** por el contenido de la celda de memoria cuya dirección es la 10.725. El resultado del producto almacenarlo en el acumulador. |
| 28 | 1.038 | **Bifurcar** a la dirección de memoria 1.038 en el caso de que el contenido del acumulador sea 0 (bifurcación condicional). |
| 33 | 28 | **Bifurcar** a la dirección de memoria 28 en el caso de que el contenido del acumulador sea una cantidad negativa (bifurcación condicional). |
| 40 | 38.413 | **Bifurcar** a la dirección de memoria 38.413 (bifurcación incondicional). |
| 52 | 117 | **Leer una ficha perforada** por la unidad de entrada y depositar el contenido de cada uno de sus diez datos en celdas de memoria (una por cada dato), consecutivamente, a partir de la dirección de memoria 117 (*). |
| 58 | 428 | **Escribir en una línea de impresora,** por la unidad de salida, el contenido de las 10 celdas de memoria que se encuentran a partir de la dirección 428 de memoria. |
| 59 | (indiferente) | **Terminar la ejecución del programa** (parada). La cantidad puesta en el campo de dirección de operando, puede ser cualquiera (**). |

### c) El programa

A continuación se va a presentar un ejemplo de programa de ordenador escrito con las instrucciones enunciadas anteriormente y con la configuración de memoria y datos también descrita.

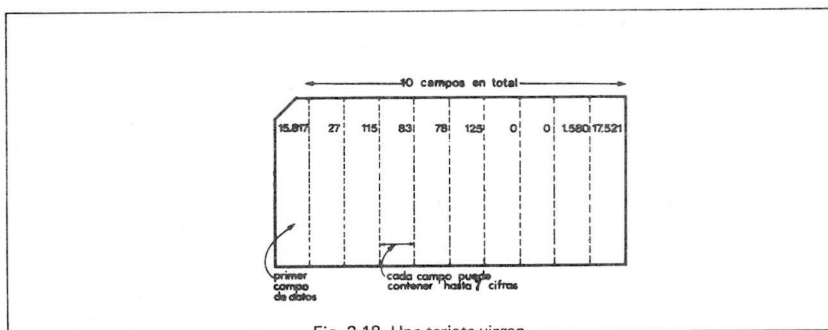



Fig. 2.18. Una tarjeta virgen.

### Lo que debe hacer el programa

El programa deberá realizar una facturación muy simple. Consiste en leer varias fichas perforadas. En cada ficha pueden venir hasta 10 datos, según se explicó anteriormente; pero para facilitar el ejemplo, consideremos que sólo existen datos en los dos primeros campos de cada ficha.

Dichos campos contendrán respectivamente:

**—Código del artículo.**
**—Cantidad de artículos pedidos.**

Simbólicamente se denominarán ART y CANT respectivamente. La figura 19 muestra el lote de tarjetas que formará la información de entrada al ordenador.

El programa mandará leer una ficha. A continuación comprobará si el código del artículo (ART) es alguno de los siguientes***

—141 (que se denominará simbólicamente ART 1)
— 32 (que se denominará simbólicamente ART 2)

*Las instrucciones de entrada y salida no son tan simples en la realidad; se ha tomado ésta tan sencilla para facilitar al lector el seguimiento del ejemplo. Se supondrá también que en cada ficha perforada pueden entrar hasta 10 cantidades o datos diferentes, cada uno de los cuales puede tener hasta 7 cifras y signo (ver fig. 18).

**Téngase en cuenta que la Unidad Central al decodificar el código de operación (59), interpreta que se trata de una instrucción de parada, por lo que le resulta indiferente la parte de dirección del operando.

***Se han figurado únicamente dos tipos distintos de artículos para facilitar al máximo el programa.

Estos datos los tendrá almacenados en memoria el ordenador como parte de una pequeña "lista de precios".

Identificado el punto anterior, deberá multiplicar la cantidad de artículos pedidos (CANT) por el precio unitario del artículo de que se trate, almacenado igualmente

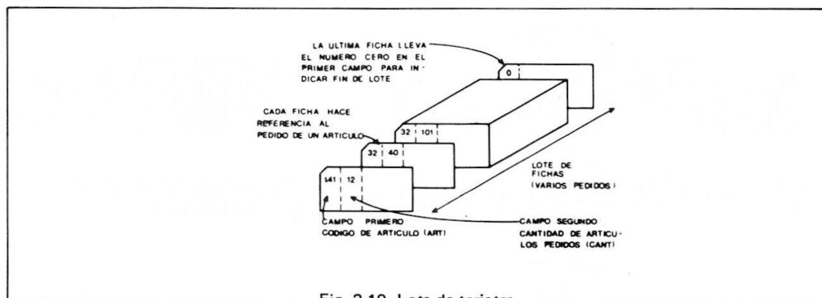



Fig. 2.19. Lote de tarjetas.

en memoria (es decir por el precio de ART 1 ó ART 2). El precio total resultante se denominará simbólicamente TOTAL. Caso de no encontrar igualdad de ART con ART 1 ni ART 2, el programa deberá ignorar la ficha y proceder a leer la siguiente del lote.

Los precios unitarios almacenados en la memoria del ordenador, son los siguientes:

–23 (precio unitario de ART 1) que se denominará de forma simbólica PRECIO 1
–42 (precio unitario de ART 2) que se denominará de forma simbólica PRECIO 2

Una vez efectuada la multiplicación, es decir una vez obtenido el precio total de los artículos pedidos en la ficha, se procederá a imprimir los resultados por la impresora. En cada línea se imprimirá el código del artículo pedido (ART), cantidad pedida (CANT) y precio que resulta (TOTAL), es decir la facturación que corresponde a la ficha de entrada. A continuación pasará a leer una nueva ficha repitiéndose todo el proceso.

El ordenador deberá terminar el programa cuando se encuentre una ficha cuyo primer campo sea cero (esta ficha especial se habrá situado previamente al final del lote de tarjetas).

A continuación se presenta en la figura 20 una secuencia de símbolos encerrados en casilleros que se denomina "diagrama de flujo del programa", o simplemente "organigrama".

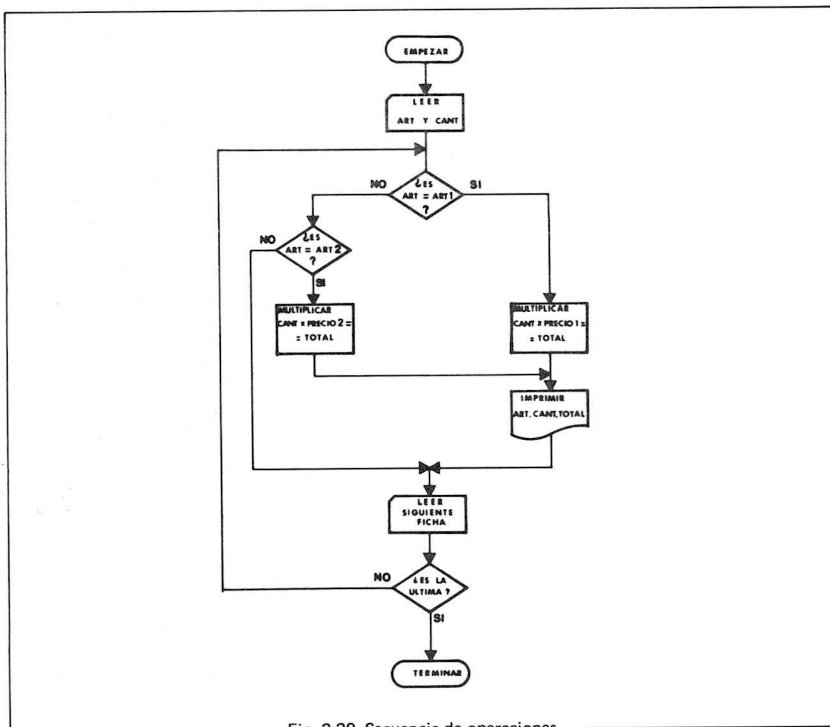



Fig. 2.20. Secuencia de operaciones.

Se trata de la representación esquemática de la secuencia de operación ya explicada anteriormente, y se utiliza mucho en la técnica de programación de ordenadores (ver capítulo 7).

**La configuración del programa y los datos en la memoria.**

La figura 21 muestra la ubicación en memoria del programa que obedece a la secuencia de operaciones de la figura 20.(*)

*El lector deberá tratar de comprobar primero y reconstruir después la secuencia de instrucciones que componen el programa.

| DIRECCIONES DE MEMORIA | MEMORIA | | COMENTARIOS |
|---|---|---|---|
| 0 | 52 | 85 | Leer una ficha. Almacenar los 2 datos a partir de la dirección 85. |
| 1 | 05 | 85 | Cargar el acumulador con el contenido de la posición 85 (ART). |
| 2 | 22 | 100 | Restar del acumulador el contenido de la posición 100 (ART1 = 141). |
| 3 | 28 | 8 | Saltar a la posición 8 si el acumulador fuese cero (ART = ART1). |
| 4 | 05 | 85 | Cargar el acumulador con el contenido de la posición 85 (ART). |
| 5 | 22 | 153 | Restar del acumulador el contenido de la posición 153 (ART2 = 32). |
| 6 | 28 | 11 | Saltar a la posición 11 si el acumulador fuese cero (ART = ART2). |
| 7 | 40 | 15 | Saltar incondicionalmente a la posición 15. |
| 8 | 05 | 86 | Cargar el acumulador con el conteni do de la posición 86 (CANT). |
| 9 | 25 | 101 | Multiplicar contenido de acumulador (cant.) por contenido posición 101 (PRECIO 1). |
| 10 | 40 | 13 | Saltar incondicionalmente a la posición 13. |
| 11 | 05 | 86 | Cargar el acumulador con el contenido de la posición 86 (CANT.). |
| 12 | 25 | 154 | Multiplicar contenido de acumulador (CANT) por contenido posición 154 (PRECIO 2). |
| 13 | 11 | 87 | Almacenar el contenido del acumulador (TOTAL) en la posición 87. |
| 14 | 58 | 85 | Escribir los contenidos de las tres posiciones a partir de la 85 (ART, CANT, TOTAL). |
| 15 | 52 | 85 | Leer una ficha. Almacenar los 2 datos a partir de la dirección 85. |
| 16 | 05 | 85 | Cargar el acumulador con el contenido de la posición 85 (ART). |
| 17 | 28 | 19 | Saltar a la posición 19 si el acumulador fuese cero. |
| 18 | 40 | 2 | Saltar incondicionalmente a la posición 2. |
| 19 | 59 | 0 | Parar la ejecución del programa. |
| ⋮ | | | |
| 85 | | | Celda para almacenar el código del artículo (ART). |
| 86 | | | Celda para almacenar la cantidad de artículos (CANT). |
| 87 | | | Celda para almacenar el precio total factura (TOTAL). |
| ⋮ | | | |
| 100 | 141 | | Celda que contiene el número 141 (CODIGO ART1). |
| 101 | 23 | | Celda que contiene el número 23 (PRECIO 1). |
| ⋮ | | | |
| 153 | 32 | | Celda que contiene el número 32 (CODIGO ART2). |
| 154 | 42 | | Celda que contiene el número 42 (PRECIO 2). |

**Fig. 2.21. Memorización secuencia fig. anterior.**

Se ha supuesto que el programa está almacenado a partir de la primera celda de memoria (posición 0). A continuación del programa irán los datos que necesite utilizar el programa y las posiciones reservadas por el mismo. Junto a cada instrucción se indica un comentario sobre la función que cumple la misma.

Se puede comprobar que el programa ocupa 20 celdas de memoria (celdas 0 a la 19) y los datos y zonas para lectura, otras adicionales.

Dado que el acumulador actúa en todo momento de intermediario para traer y llevar datos de memoria, hay que restituir frecuentemente sus anteriores contenidos, puesto que son destruidos continuamente.

**Los resultados**

Para finalizar este apartado se presentan esquemáticamente los resultados obtenidos al ejecutar el programa ante unos datos de entrada concretos.

La figura 22 muestra el proceso general de la ejecución del programa y los resultados que presentaría.

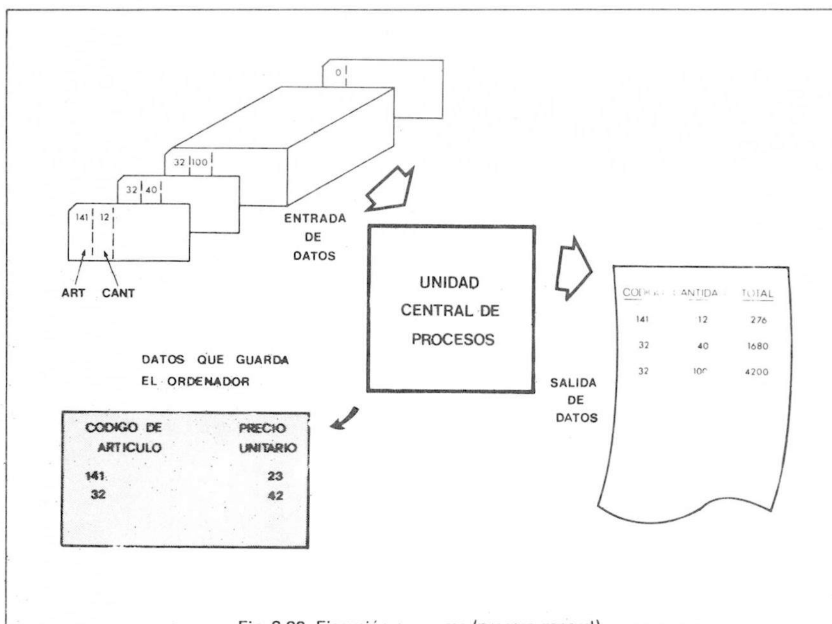



Fig. 2.22. Ejecución programa (proceso general).

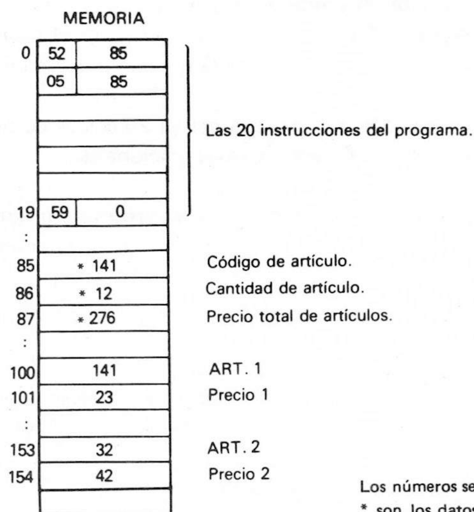



Fig. 2.23. Instantánea de la memoria.

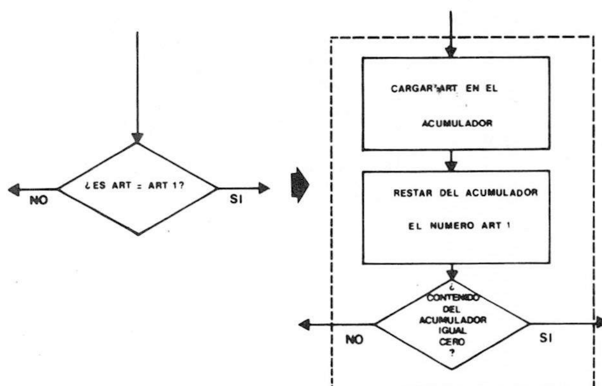



Fig. 2.24. Comparación de dos cantidades.

Para detallar un poco más el proceso de la ejecución del programa, se presenta en la figura 23 una instántanea de la memoria al terminar de ejecutarse la primera facturación, es decir después de procesar los datos de la primera ficha.

La figura 24 viene a explicar las instrucciones que hacen falta para efectuar una comparación de dos cantidades (ART y ART 1, por ejemplo), dado que en el repertorio no existe una instrucción que la realice directamente. Es un ejemplo característico de la necesaria manipulación de instrucciones, al diseñar un programa, cuando la estructura de aquéllas es tan simple que no permiten efectuar ciertas operaciones con una sola de ellas, ni aun siendo dichas operaciones tan elementales como una simple comparación de dos cantidades.

### 2.3.5. Interrupciones

Una interrupción es una señal que permite a la Unidad Central atender en su momento cualquier posible eventualidad que pueda ocurrir durante la ejecución del trabajo normal.

Una interrupción puede producirse por diversas causas, provocando acciones distintas en cada caso. De acuerdo a ello, dividiremos las interrupciones, entre otros tipos, en:

- **De entrada-salida.** Informan de la terminación de una operación de entrada-salida por un periférico (Canal) quedando éste disponible.
- **De programa.** Informan de una anormalidad detectada en el desarrollo de un programa.
  - Instrucciones inexistentes
  - Instrucciones no permitidas
  - Acceso a zonas de memoria protegidas
  - Direccionamientos inexistentes
  - Overflow
  - etc.
- **Por errores de máquina.** Informan de errores Hardware.
  - Errores de paridad
  - Transferencias de información erróneas
  - Fallos en la alimentación
  - etc.
- **Externas.** Provocadas por elementos externos a la CPU.
  - Por el operador
  - Por el reloj, cada cierto tiempo
  - Interrupciones en prueba (**debugging**)
  - etc.

Cuando se produce una interrupción, el circuito que la genera activa algún bit de ciertos registros especiales de la memoria, cediendo control, tras haber salvado el **contador de instrucciones**, a los programas de control, que investigarán la causa y llevarán a cabo las acciones requeridas en cada caso.

A fin de prever la ocurrencia simultánea de varias interrupciones, suelen establecerse entre ellas **niveles de prioridad**, de tal forma que se atiendan primero las más prioritarias. Por otra parte, suele haber también **colas de espera** donde irán a parar las interrupciones del mismo nivel que se vayan produciendo, que son atendidas por su orden de ocurrencia.

Las interrupciones suelen atenderse por lo general cuando acaba de ejecutarse totalmente una instrucción, si bien algunas, de máxima prioridad, pueden cortar dicha ejecución. Una vez atendida una interrupción, y si no hay otras pendientes, la UCP cederá control al proceso que se estaba ejecutando al producirse la interrupción, en el punto en que se quedó, recuperando el contador de instrucciones.

Una aplicación muy usual del sistema de interrupciones se da en los equipos de Tiempo Compartido, en los que un reloj genera una interrupción cada cierto tiempo, momento en el cual la UCP cederá control al proceso del siguiente terminal.

## CAPITULO 3. DISPOSITIVOS PERIFERICOS

### 3.1. Introducción. Soportes de información y tipos de periféricos

En la figura 1 se presenta un esquema sobre la evolución que experimenta la información desde su nacimiento hasta que es procesada por el ordenador y aparece por las unidades de salida constituyendo los resultados del proceso.

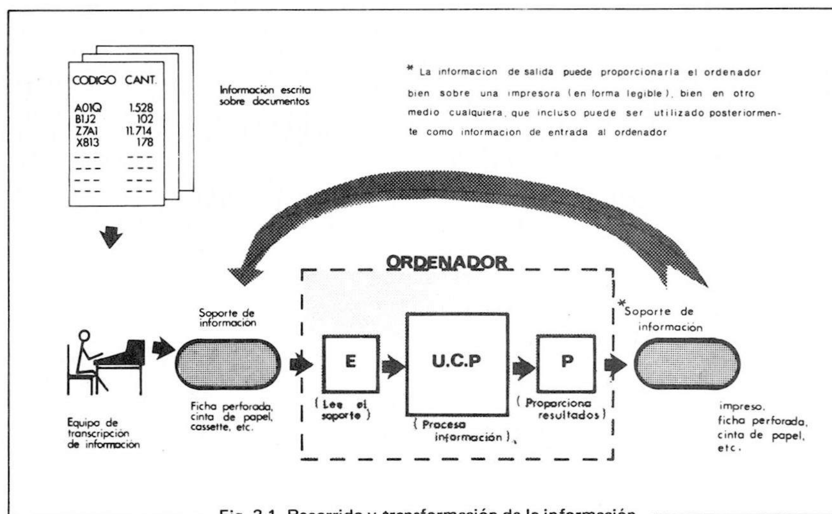



Fig. 3.1. Recorrido y transformación de la información.

La figura muestra la existencia de elementos marcados E y S, aparte de la U.C.P. que ya se ha estudiado. Estos elementos (Entrada y Salida) representan los dispositivos de comunicación del ordenador con su entorno es decir, los medios por los que se proporciona información al ordenador y aquellos otros por los que el ordenador suministra información de salida.

A todas las unidades que se encargan del trasiego de la información entre el exterior y la Unidad Central o viceversa se las denomina unidades o elementos **periféricos** (o simplemente periféricos).

Un primer concepto al hablar de la información es el **soporte** o medio físico sobre él que va dispuesta. Normalmente el soporte primario de aquella puede ser una serie de documentos legibles, es decir, confeccionados mediante los caracteres ordinarios (por ejemplo albaranes con los pedidos a una empresa; documentos en que figuran los días y horas trabajadas de los empleados de una oficina, etc.). En principio podría imaginarse como ideal que los órganos o periféricos de entrada del ordenador, pudiesen leer directamente estos documentos, pero esto no es posible normalmente.

Por lo tanto la información primaria (cuyo soporte eran hojas de papel) se le somete a un proceso de **transcripción** a otro soporte que sea "legible" por el ordenador, utilizando algún equipo transcriptor. De esta manera, cada carácter o signo legible de la información primaria se transcribe, con arreglo a unos determinados códigos, en forma de perforaciones sobre una pequeña cartulina o bien sobre un material imanable a base de diminutos puntos marcados, etc, etc.

La información sobre este nuevo medio o soporte, puede ser leída ahora por los órganos de entrada del ordenador. Existen periféricos de entrada especializados en leer cada tipo diferente de soporte.

La información una vez introducida en la Unidad Central estará en forma binaria (unos y ceros), que es aquella con la que puede trabajar internamente el ordenador.

De la misma forma, la información de salida se podrá suministrar por el ordenador en un variado repertorio de soportes. La mayoría de las veces interesa que los resultados de los trabajos que ejecuta el ordenador, aparezcan en forma legible, es decir, en información escrita sobre papel; otras veces se suministra en otros soportes similares a los utilizados para la entrada de información, que puedan ser nuevamente leídos por el ordenador (reciclaje de la información).

Los periféricos de salida igualmente pueden ser de diversos tipos obedeciendo al medio o soporte en que deban proporcionar la información de salida.

Los soportes más utilizados son:

- Tarjetas perforadas.
- Cinta perforada.
- Cinta magnética.
- Discos y tambores magnéticos.
- Discos flexibles o «diskette».
- Papel continuo.

Todos ellos se estudiarán en detalle más adelante.

Atendiendo a la forma de acceso a los datos, los soportes pueden ser:

- De acceso secuencial.
- De acceso directo.

Una cinta magnética es un soporte de **acceso secuencial**, en cuanto que un dato grabado en su mitad no es accesible sino por la lectura previa de todos los anteriores desde el comienzo del carrete.

Un soporte de **acceso directo**, como puede ser un disco magnético y sobre todo la memoria, nos permite acceder a cualquier dato grabado en él directamente, calculando el lugar donde está ubicado dentro del disco y yendo directamente a ese lugar.

En cuanto a los tipos de dispositivos periféricos, se distingue entre periferia **local** y periferia **remota**; la primera está constituida por los distintos elementos periféricos conectados a pocos metros de la Unidad Central (el conjunto Unidad Central y periferia local, se denomina corrientemente "Subsistema del Ordenador Central"); los periféricos "remotos" pueden estar situados a distancias de kilómetros, estando conectados al ordenador por algún **medio de comunicación** (por ejemplo línea telefónica).

Por otra parte, podemos agrupar los dispositivos en tres grupos, de acuerdo a su utilización:

- De Entrada.
- De Salida.
- De Entrada/Salida (almacenamiento).

Dispositivos de Entrada serían aquellos que sirven para introducir información al ordenador. De Salida, aquellos por los cuales el ordenador proporciona información. De E/S son los que se utilizan para ambas cosas. En este sentido pueden ser llamados dispositivos "de almacenamiento", o memorias externas, en cuanto que el ordenador "almacena" en ellos datos que puede recuperar en cualquier momento; esta concepción se debe limitar, no incluyendo en ella soportes como las tarjetas perforadas, que a pesar de ser utilizables como entrada y también para salida, no son "reutilizables", ya que no se puede cambiar un dato ya perforado.

## 3.2. Periféricos

### 3.2.1. La tarjeta perforada

Fue ideada en el año 1887 por Hollerit en los Estados Unidos. Ha sido y es sin duda el soporte de información más extendido en su empleo. El modelo más normal es una cartulina de 83 x 188 mm. Está estructurada en 80 **columnas** verticales y 12 **filas** horizontales. La forma habitual de representar la información es transcribir un carácter en cada columna a base de una determinada combinación de, perforaciones debidamente situadas en la citada columna de la tarjeta.

Fig. 3.2. Tarjeta perforada.

La figura muestra una tarjeta perforada en la que aparecen los caracteres más usuales con las distintas combinaciones de perforación que le corresponden.

A la tarjeta perforada se la denomina **registro unitario** puesto que se lee o perfora habitualmente como una unidad de información (datos de un albarán, información sobre un empleado, etc.); a efectos funcionales se la divide en distintos **campos** de información (por ejemplo, la cantidad y el número de artículos de un albarán constituirían dos campos diferentes), comprendiendo cada campo un número determinado de columnas.

En la parte superior de la tarjeta pueden aparecer escritos los textos descriptivos del contenido de los campos, para su más fácil identificación.

Las lectoras primitivas de tarjetas constaban básicamente de los siguientes órganos, actuando en la secuencia siguiente:

- Un casillero de entrada donde se sitúan las tarjetas a leer.
- Un dispositivo de arrastre para transportar tarjeta a tarjeta a un banco de lectura.
- Un banco de pivotes sujetos por unos muelles. Existían tantos pivotes como posibles perforaciones podían encontrarse en la tarjeta, normalmente 960; de esta forma, cuando la tarjeta se detenía en este banco de pivotes se soltaba el conjunto de los muelles, y los pivotes que coincidían con una perforación cerraban un circuito, permaneciendo el resto de pivotes comprimidos sobre sus muelles sin atravesar las tarjetas.
- Un conjunto de circuitos, citados anteriormente, cuya situación, con ayuda de los pivotes, representaba una "imagen eléctrica" de la información leída.
- El mismo dispositivo de arrastre que actuó a la entrada movía las tarjetas ya leídas a un casillero de salida.

Este juego de pivotes era lento y poco seguro, por lo cual se sustituyó, en una etapa siguiente, por 960 grupos de escobillas; sin embargo, este sistema continuaba siendo lento, por ser obligada la detención en el banco de lectura propiamente dicho.

Más adelante, estos equipos evolucionaron al aparecer el método de "escobilla-tambor", que es hoy día utilizado muy frecuentemente; al igual que en el procedimiento primitivo, existen unos dispositivos de arrastre, pero la tarjeta no realiza parada en ningún momento. La innovación consiste en reemplazar el banco de lectura por un cilindro metálico de contacto y 80 escobillas, tantas como columnas.

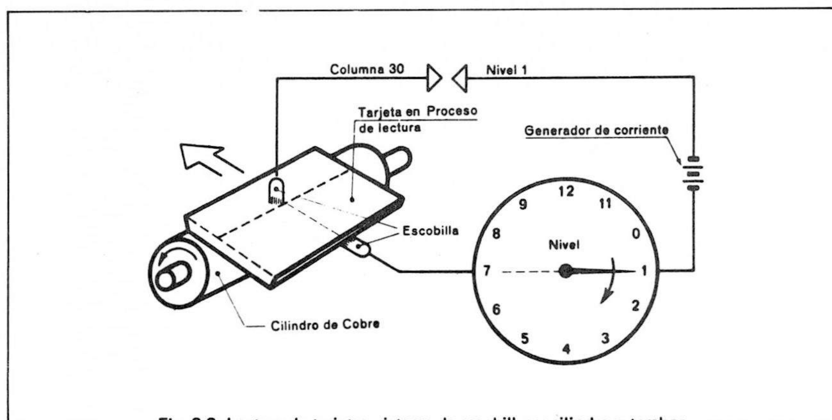



Fig. 3.3. Lectora de tarjetas, sistema de escobillas y cilindro o tambor.

En la figura se observa el esquema de la operación; el movimiento de la tarjeta es simultáneo y coordinado con un sistema cronometrador que mecánicamente está acoplado al cilindro, aunque para mayor claridad de la figura ha sido dibujado independiente, el cual establece en niveles las distintas posiciones (zona 12, zona 11, zona 0, dígito 1, ... dígito 9), que estamos leyendo en cada momento, ya que existen exclusivamente 80 escobillas que leen paso a paso las 12 filas de la tarjeta. En este sistema se sigue aprovechando el principio de obtener una "imagen eléctrica" de la información leída.

En otras lectoras se emplean 12 escobillas —tantas como filas—, y la lectura de la tarjeta se realiza columna a columna; para aumentar la seguridad de estos equipos suelen utilizarse dos estaciones de lectura, de tal manera que la segunda estación de lectura sirve como confirmación de la operación efectuada en la primera estación.

Aunque el método expuesto anteriormente continúa usándose, algunas lectoras, en vez de emplear el dispositivo de "escobilla-tambor", utilizan un juego de células fotoeléctricas y un foco luminoso, en cuyo caso la ausencia o existencia de perforación interrumpe o permite el paso de la luz, con lo cual nuevamente tenemos una "imagen eléctrica" de la información en los circuitos acoplados al juego de 80 células fotoeléctricas. Aunque el "tiempo de respuesta" eléctrica de una célula fotoeléctrica es muy pequeño, sin embargo los procedimientos mecánicos de movimiento de la tarjeta condicionan la velocidad de lectura como consecuencia de su relativa lentitud.

La velocidad de las lectoras existentes en la actualidad suele oscilar entre 100 y 2.000 tarjetas por minuto.

Para obtener los resultados de un proceso, perforados sobre tarjetas, lo cual puede interesar en ciertos casos particulares en que estos datos vayan a ser tratados posteriormente, se conecta al ordenador un elemento o unidad denominado perforador/a de tarjetas.

Básicamente, esta máquina está constituida por:

- Un casillero de entrada donde se sitúan las tarjetas que van a ser perforadas, por consiguiente sin ninguna información.
- Un dispositivo de arrastre para trasladar las tarjetas.
- Una estación de perforación formada por 80 ó 12 punzones afilados, según la tarjeta vaya a ser perforada por filas o columnas, cuyo movimiento está gobernado de acuerdo con la información que vamos a perforar.
- Una estación de lectura que al realizar la lectura de la información perforada sirve como confirmación de que la operación se ha realizado correctamente.
- Un casillero de salida para recoger las fichas ya perforadas.

Al igual que en la lectora de tarjetas, la operación de perforación se hace línea por línea, o columna por columna, y simultáneamente en las 80 columnas o 12 filas. La velocidad de perforación de los elementos actuales oscila entre 50 y 500 tarjetas por minuto, lógicamente inferior a la velocidad de las lectoras de tarjetas por intervenir una operación de perforación, que en cualquier caso es relativamente muy lenta.

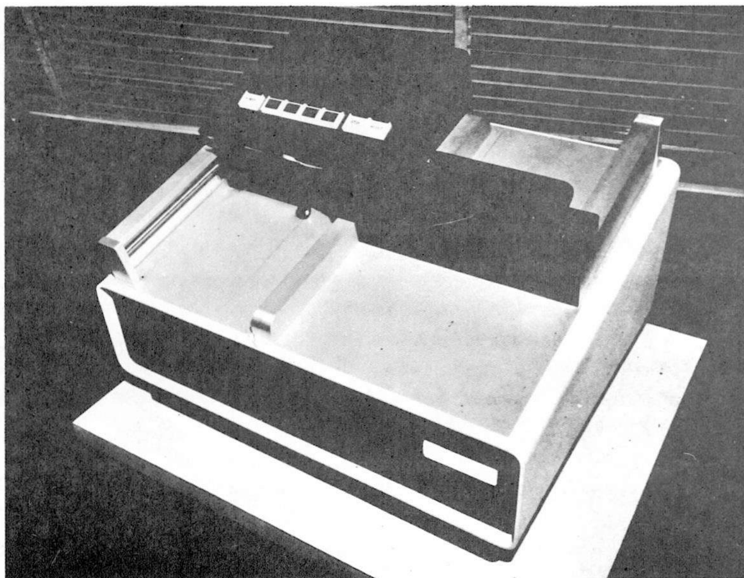


Foto 3.1. Lectoras de tarjetas perforadas.

### 3.2.2. Cinta perforada

Este soporte no pertenece en exclusiva al campo de los ordenadores, puesto que se ha venido utilizando en equipos de telegrafía y transmisión de datos; consiste en una cinta de papel en la que pueden efectuarse una serie de perforaciones circulares siguiendo unas líneas prefijadas paralelas a los bordes de la cinta llamadas **canales.**

Para representar la información en estas cintas, cada carácter se transcribe mediante un juego de perforaciones en sentido vertical, pudiendo perforarse opcionalmente cada uno de los canales. Existen cintas perforadas de cinco, seis, siete u ocho canales.

Independientemente del número de canales de la cinta, existe siempre un canal llamado de **arrastre** con perforaciones continuas más pequeñas, que únicamente se emplea a efectos mecánicos, para hacer avanzar la cinta.

La cinta perforada es un soporte continuo, por lo cual puede contener información de cualquier longitud, y desde este punto de vista aventaja a la tarjeta perforada al permitir un aprovechamiento superior. Sin embargo no tiene la comodidad de la tarjeta al no poder reemplazarse mediante una simple sustitución, un registro unitario por otro. Para su almacenamiento o envío por correo, por ejemplo, se puede enrollar sobre un carrete y ocupa un volumen relativamente reducido. La figura muestra una porción de cinta de papel perforada.

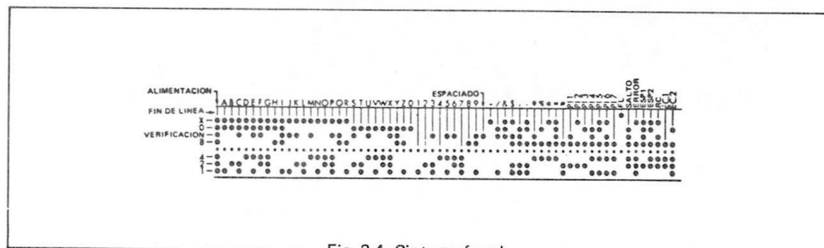



Fig. 3.4. Cinta perforada.

La introducción de la información contenida en las cintas perforadas en la memoria del ordenador se realiza con el lector de cinta perforada. El principio de los lectores de cinta perforada es análogo al sistema de lectura de un teletipo, con la diferencia de que un lector lee a un mismo tiempo todos los elementos codificados que corresponden a un carácter y los transmite en paralelo, carácter a carácter, al ordenador, mientras que los sistemas de los teletipos leen en paralelo, pero transmiten en serie, elemento a elemento.

Un sistema de lectura —poco generalizado— en estas lectoras se basa en el mismo principio que el de la lectora de tarjetas, mediante un juego de escobillas; en función de la existencia o ausencia de perforaciones se establece una "imagen eléctrica" del carácter que corresponde a la configuración leída. La lectura se hace

carácter a carácter, por columnas, haciendo avanzar la cinta mediante una rueda que engarza en las perforaciones de arrastre.

El procedimiento comúnmente empleado consiste en sustituir el juego de escobillas y los contactos correspondientes por una fuente luminosa y un banco de fotodiodos. Estos fotodiodos son unos dispositivos sensibles a la luz que presentan gran resistencia cuando no están excitados. Al presentarse una perforación la luz excita al diodo, cerrándose con ello un flujo de corriente que indica la existencia de la perforación.

La velocidad de las lectoras oscila entre 100 y 2.000 caracteres por segundo.

La perforación de las cintas de papel, en forma análoga a las tarjetas, puede ser realizada on u off-line. La perforación on-line se realiza por medio de una unidad conectada al ordenador denominada perforador de cinta.

Estos equipos constan esencialmente de una estación de perforación, donde se efectúan las perforaciones mediante punzones afilados, carácter a carácter, y de una estación de lectura, generalmente por fotodiodos, para confirmar que la operación realizada es correcta.

La velocidad de estos elementos suele oscilar entre 50 y 300 caracteres por segundo.

Para la perforación off-line existen numerosos equipos auxiliares que generan cinta de papel perforada capaz de ser leída por un lector de cinta, tales como teletipos, máquinas fotocomponedoras, etc.

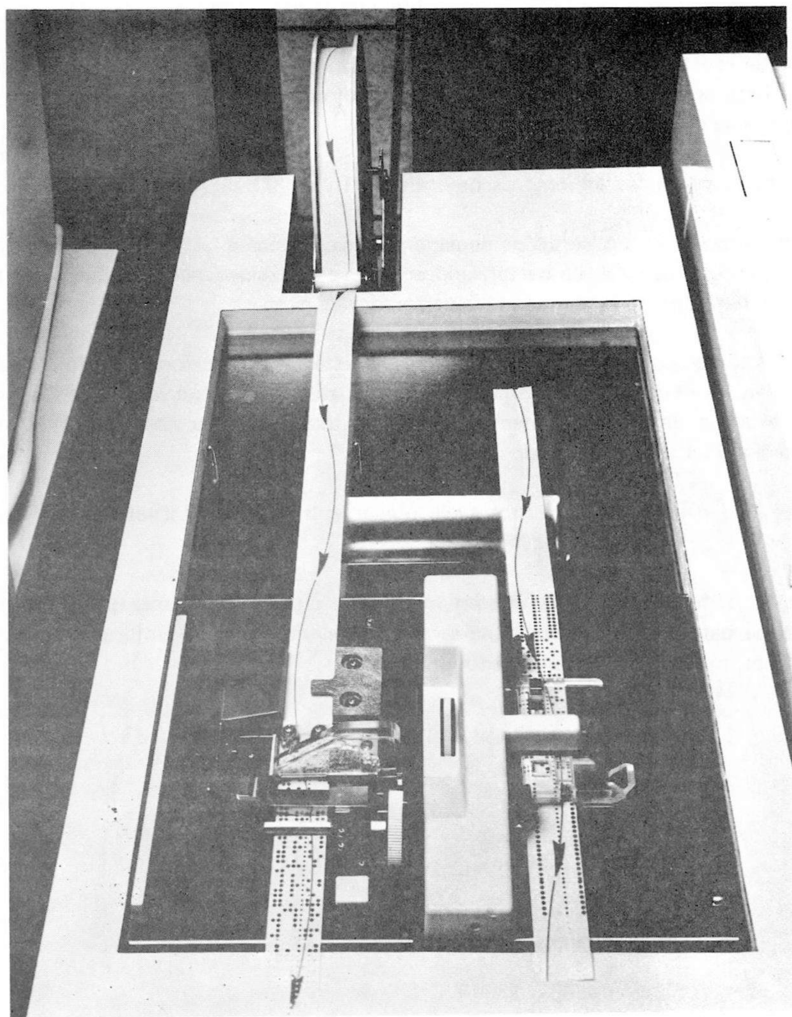


Foto 3.2. Lector-perforador de cinta de papel

### 3.2.3. Cinta magnética

Está constituida por un substrato (generalmente un poliéster) sobre el que hay una segunda capa de material magnético (óxido de hierro). Sus longitudes más usuales suelen ser de 600, 1.200 y 2.400 pies; (un pie = 0,3048 metros); el ancho de estas cintas es de 13 mm.

La información queda registrada en imanaciones locales sucesivas de la superficie (cada **punto o imán elemental** representa un bit de información), que forman **pistas** paralelas en sentido longitudinal de la cinta. Existe un paralelismo entre el canal de la cinta de papel y la pista de la cinta magnética y en la forma de presentar la información, grabando un carácter en cada columna teórica, de forma transversal. Se emplean generalmente cintas de siete o nueve pistas (ver Figura 5).

Un parámetro importante en las cintas magnéticas es la **densidad de grabación**, que es la cantidad de información que se graba por unidad de longitud; se suele

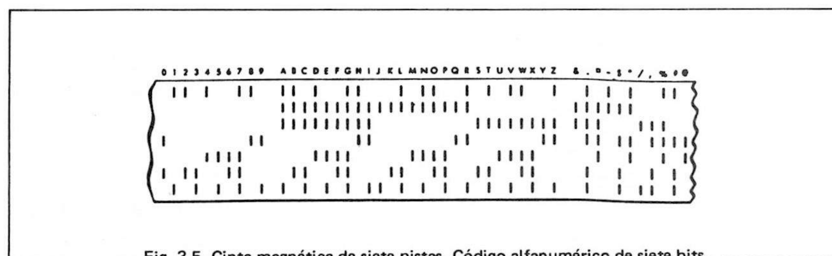


Fig. 3.5. Cinta magnética de siete pistas. Código alfanumérico de siete bits.

expresar en número de columnas/pulgada o lo que es lo mismo en bits/pulgada de pista (b.p.i.) (bits/inch o bits/pulgada, 1 pulgada = 2,54 centímetros). Las densidades de grabación más corrientes son 800 y 1.600 b.p.i.

La información contenida en las cintas se organiza en **bloques.** Cada bloque contiene una o varias informaciones unitarias (recuérdese el concepto de registro unitario expuesto al tratar sobre las fichas perforadas) y está separado de otro por un segmento de cinta sin información.

La capacidad de una cinta puede ascender a varios millones de caracteres (compárese con la capacidad de la memoria central que era de varios miles de caracteres).

Para su manejo y transporte las cintas se enrollan en unos carretes, al igual que se hacia con las cintas de papel.

A diferencia de los soportes ficha y cinta de papel, en la cinta magnética, discos, tambor, un mismo equipo conectado al ordenador puede realizar operaciones de entrada y salida, o, lo que es lo mismo, de lectura y grabación.

El principio de grabación y de lectura es similar asimismo para cintas, discos y

tambores. Se basa en la existencia de uno o varios juegos de dos cabezas, una de grabación y otra de lectura, constituidas una y otra en esencia por un electroimán.

Para la operación de grabación se hace pasar por el devanado del electroimán de la cabeza de grabación un impulso de corriente. Si la superficie magnetizable de la cinta, disco o tambor esta situada muy próxima, se inducirá en un punto de ella un campo magnético de un sentido u otro, según cual sea el sentido de la corriente del devanado, lo que representará un bit de información.

Para la operación de lectura se hace pasar la superficie grabada anterior por delante de la cabeza de lectura, induciéndose en el devanado una corriente de un sentido u otro, según sea el sentido de imantación del punto que pasa por delante de la cabeza.

Las unidades de cinta magnética poseen un juego de cabezas de lectura y escritura por cada pista, realizándose, tanto la lectura como la grabación, en paralelo en todas ellas.

Además de las cabezas de grabación y lectura, una unidad de cinta magnética está constituida básicamente por: un carrete proveedor de información, que desenrrolla la cinta, otro receptor encargado de enrrollarla y una unidad de tensión y movimiento.

Para evitar los esfuerzos mecánicos sobre la cinta la unidad de tensión y movimiento tiene a cada lado de las cabezas lectoras-grabadoras cuatro cámaras, de las cuales las dos superiores están a la presión normal y las inferiores a un vacío suficiente para que cierta longitud de cinta se asiente cómodamente sin ninguna tensión; de esta forma, la parte de la cinta que pasa por las cabezas procede de una cámara inferior de vacío, por lo que no requiere prácticamente ningún esfuerzo; igual sucede con la cinta que sale procedente de las cabezas. El giro de los carretes se gobierna mediante el control de la longitud de la cinta de las cámaras inferiores de vacío.

Una vez que ha comenzado la lectura o grabación de un bloque, la velocidad con que los caracteres se leerán o grabarán, es decir, la **velocidad de transferencia**, $V_T$, vendrá condicionada por la velocidad de arrastre $V_A$ con que la cinta pasa por delante de las cabezas y por la densidad de grabación de la cinta d. Resulta inmediato deducir que, empleando las unidades adecuadas,

$$V_T = V_A \times d$$

Las velocidades de transferencia de las unidades más corrientes oscila entre 40 y 200 K caracteres (o K octetos si la cinta es de 9 pistas) por segundo.

Para la detección del principio y final físicos de la cinta existen unas células fotoeléctricas que detectan las marcas reflexivas. El principio de funcionamiento básico de estas unidades se comprende fácilmente examinando la figura.

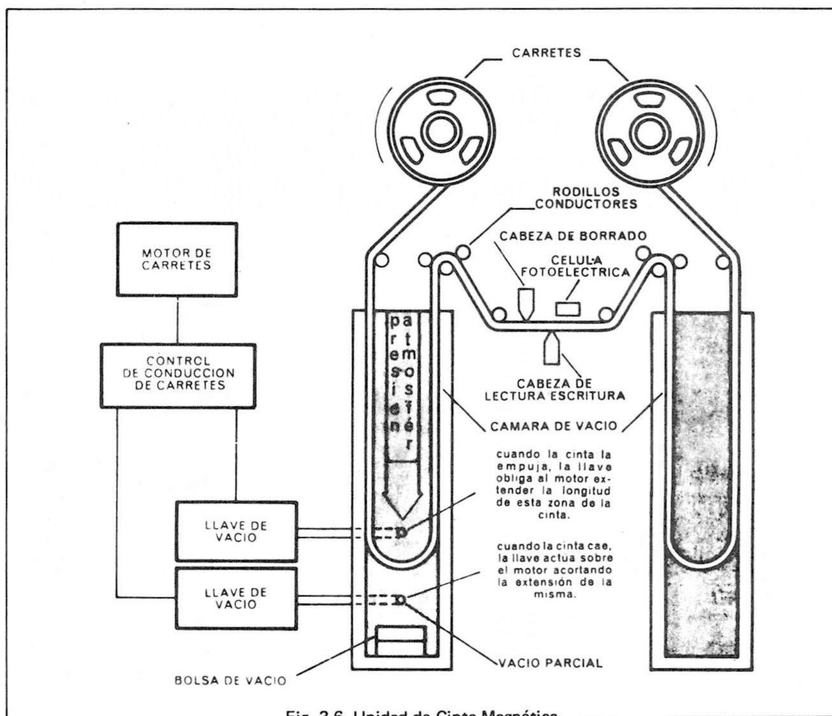



Fig. 3.6. Unidad de Cinta Magnética.

Algunas unidades de cinta magnética permiten la lectura en su recorrido inverso, lo cual tiene gran aprovechamiento en algunos tipos de aplicaciones. No es frecuente, por el contrario, la grabación en ambas direcciones.

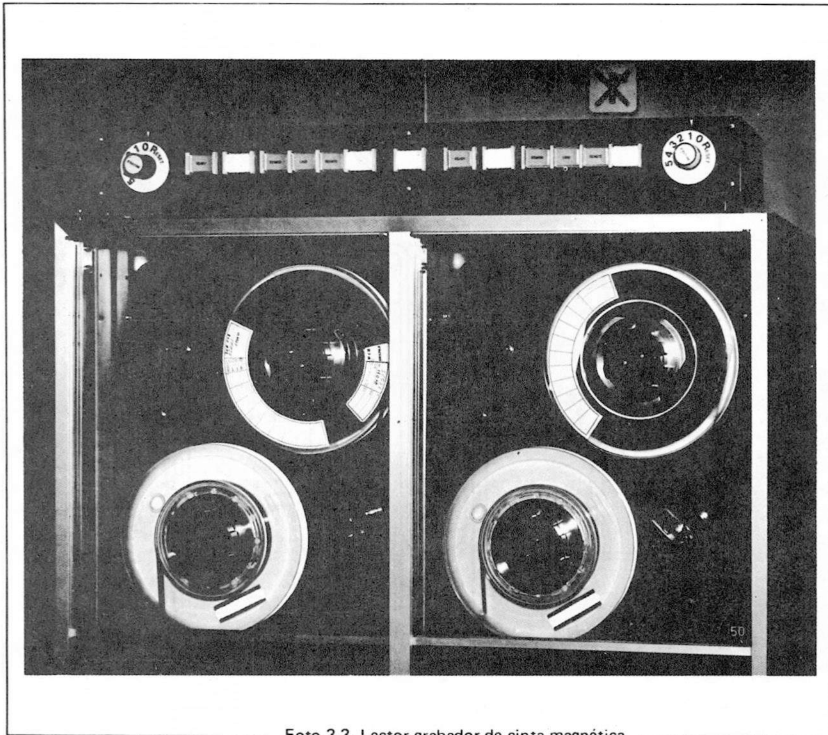



Foto 3.3. Lector-grabador de cinta magnética

### 3.2.4. Cassette

Existe una variante de cintas magnéticas, algo más reducidas en longitud y anchura y situadas en unos cartuchos exactamente iguales a los denominados universalmente "cassettes", denominándoseles por el mismo nombre en las aplicaciones de informática. Dichas cintas suelen ser de 200 ó 300 pies de longitud y la densidad de grabación de 800 b.p.i., en una cassette se puede almacenar del orden de 350.000 caracteres, y la grabación se hace longitudinalmente, bit a bit.

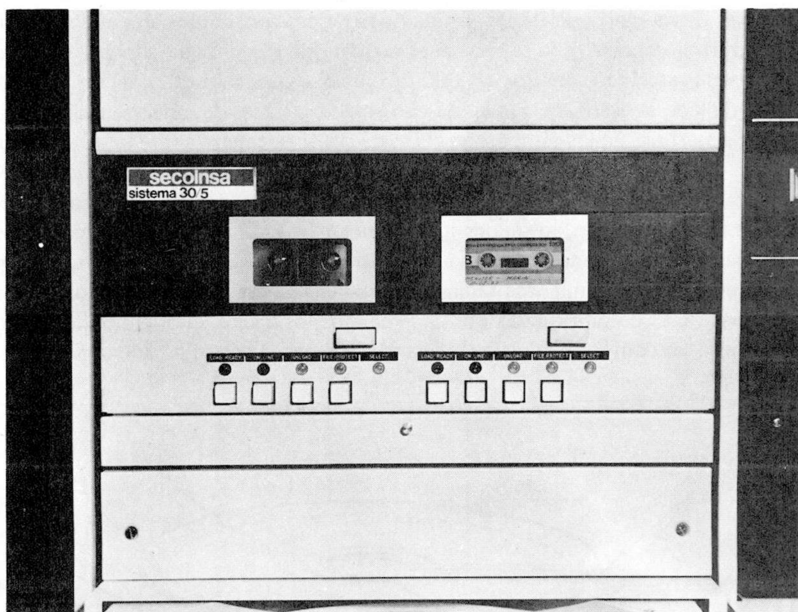



Foto 3.4. Lector-grabador de cassette.

### 3.2.5. Disco magnético

El disco magnético, como soporte de información está constituido por un disco metálico, generalmente de aluminio, cuyas dos superficies principales están recubiertas de una sustancia magnetizable. Los datos se almacenan en serie, bit a bit, magnetizando puntos sucesivos sobre alguna de las circunferencias concéntricas denominadas **pistas**, contenidas en ambas caras del disco.

Normalmente los discos son agrupados, calados a un eje común, formando lo que se conoce como un **disk-pack** (paquete de discos).

La unidad de memoria de discos, unidad periférica, consta fundamentalmente de un eje central sobre el que se monta el paquete de discos, que gira permanentemente a una velocidad superior a 1.000 revoluciones por minuto, y de un juego de cabezas de lectura-escritura, fijadas a unos **brazos**, capaces de moverse radialmente sobre la superficie de los discos, con el fin de situar la cabeza sobre la pista en que se quiere grabar o leer, según puede apreciarse en la figura. Normalmente existen tantas cabezas de lectura-escritura como superficies de discos con información contiene el disk-pack, o, lo que es igual, una cabeza por cada cara de los discos. Sin embargo, en ocasiones, cuando el diámetro de los discos es muy grande y/o se quieren evitar desplazamientos radiales demasiado largos a los brazos porta-cabezas, se sitúa más de una cabeza por cada cara de los discos, cubriendo, por tanto, cada cabeza un conjunto de pistas formando una corona circular. Incluso en algún

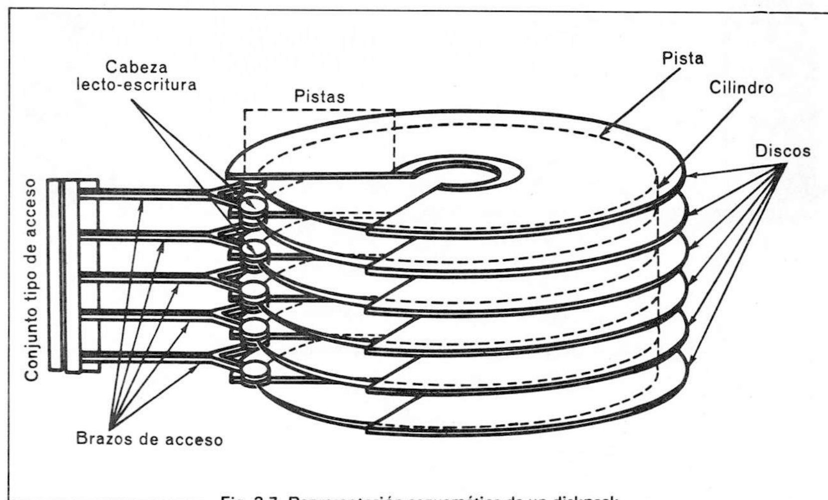



Fig. 3.7. Representación esquemática de un diskpack.

tipo de discos muy especial —utilizados, por ejemplo, en funciones de memoria central en pequeños ordenadores de oficina— existe una cabeza por cada pista. En cada operación de escritura o lectura sólo una cabeza del conjunto total está activada. Todos los brazos portacabezas están fijos a un soporte único, por lo que su movimiento es solidario, alcanzando todos a un tiempo idéntica posición radial, situándose las cabezas sobre pistas semejantes en las distintas caras de los discos que componen el disk-pack.

Se denomina **cilindro** el conjunto de todas las pistas que se pueden leer o grabar en una posición determinada de las cabezas lectoras-grabadoras. Un cilindro constará, por tanto, de tantas pistas por cara de disco como cabezas lectoras-grabadoras existan en los brazos de acceso.

Existen unos modelos de unidades de disco en las que el disk-pack es fijo; sin embargo, lo corriente es que los disk-pack sean **intercambiables**, pudiéndose ir sustituyendo a medida que son utilizados.

La memoria de discos es una memoria **direccionable** (de acceso al azar), lo que implica que, conocida la dirección donde se encuentra determinada información, puede accederse directamente a la misma sin necesidad de leer las informaciones que la preceden, tal como era necesario, por ejemplo, en las cintas de papel o magnética.

El **tiempo de acceso** a una determinada información de una memoria de discos, de la que se conoce su dirección —en principio, la pista en que se encuentra— es muy bajo, del orden de unos milisegundos (nótese, sin embargo, que ese tiempo es del orden de mil veces más lento que el correspondiente a las memorias de ferritas, semiconductores, etc.). Este tiempo de acceso puede descomponerse esencialmente en tres partes: posicionamiento, selección de cabeza y espera de rotación.

El tiempo de posicionamiento es el necesario para posicionar el conjunto de brazos de acceso, en la posición radial correspondiente a la pista buscada (recuérdese que el conjunto de brazos se movía solidariamente). Este tiempo, para una memoria de discos dada, es variable y depende lógicamente de la magnitud del desplazamiento preciso, pudiendo llegar a ser nulo si el conjunto de brazos se encuentra antes de la lectura en la posición precisa; no obstante, suele ser, por término medio, el más elevado de los componentes del tiempo total de acceso.

El tiempo de selección de cabezas, despreciable con relación a los otros dos, es el necesario para activar de entre el conjunto total de cabezas de lectura aquella que va a leer la pista deseada.

El tiempo de espera de rotación es el que transcurre desde el momento en que, seleccionada la cabeza, pasa delante de la misma el comienzo de la información a leer. Será, por término medio, el tiempo de una semirrotación (dado que los

discos son generalmente direccionables por pistas, el comienzo de la información coincide con el comienzo, entendido en sentido lógico, de la pista. No obstante, existen discos direccionables por **sectores** —1 pista = n sectores.

Una vez situada la cabeza ante el comienzo de la información a leer, comienza la transferencia hacia la memoria. Su velocidad dependerá de la velocidad con que la pista pasa delante de la cabeza de lectura (dependiente a su vez de la velocidad de rotación de los discos y del radio de la pista) y de la densidad de grabación. La velocidad de transferencia es constante, lo que implica que, al ser el radio variable, la densidad de grabación debe ser diferente según la pista, y lógicamente creciente hacia el interior del disco.

Como resumen final indicamos los valores entre los que suelen oscilar las características principales de las memorias de discos más usuales:

- Capacidad del disk-pack: 0,5 a 50 millones de caracteres.
- Tiempo medio de posicionamiento: 30 a 75 milisegundos.
- Espera media de rotación: 8 a 12,5 milisegundos.
- Velocidad de transferencia: 100 a 800 K caracteres por segundo.

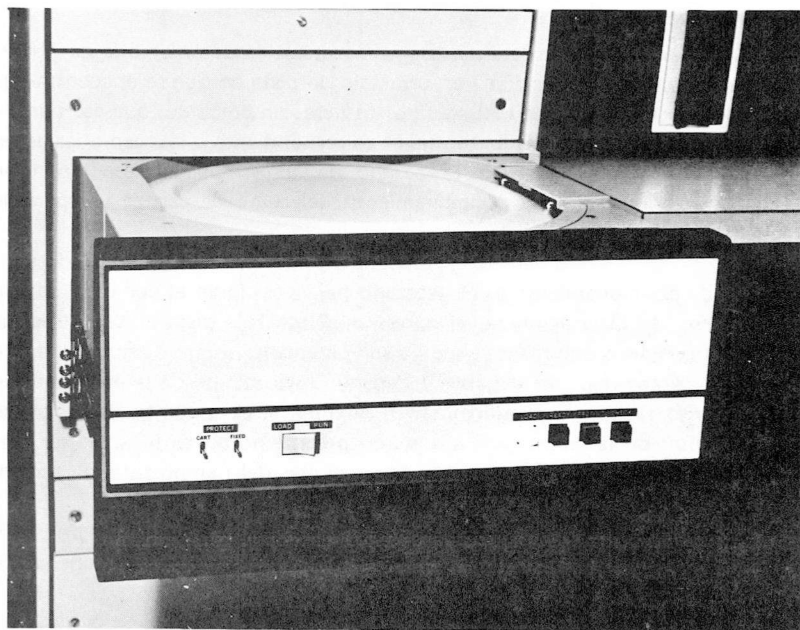



Foto 3.5. Unidad de discos

### 3.2.6. Tambor magnético

El tambor magnético, soporte de la información, está constituido por un cilindro recubierto de una sustancia magnetizable. Como en el caso del disco magnético, los datos se almacenan en serie, bit a bit, magnetizando puntos sucesivos sobre alguna de las circunferencias paralelas de la superficie (ver figura), que igualmente se denominan **pistas**.

La unidad periférica, precisa para leer y grabar tambores magnéticos, siempre fijos (no intercambiables), está constituida básicamente por un dispositivo de arrastre, que mantiene el tambor permanentemente en rotación, y un juego de cabezas de grabación y lectura similares a las existentes en cintas y discos. Prácticamente todos los modelos de memorias de tambor tienen un juego de cabezas de grabación-lectura por pista; solamente una de estas cabezas de un juego estará activa en una operación de grabación o lectura.

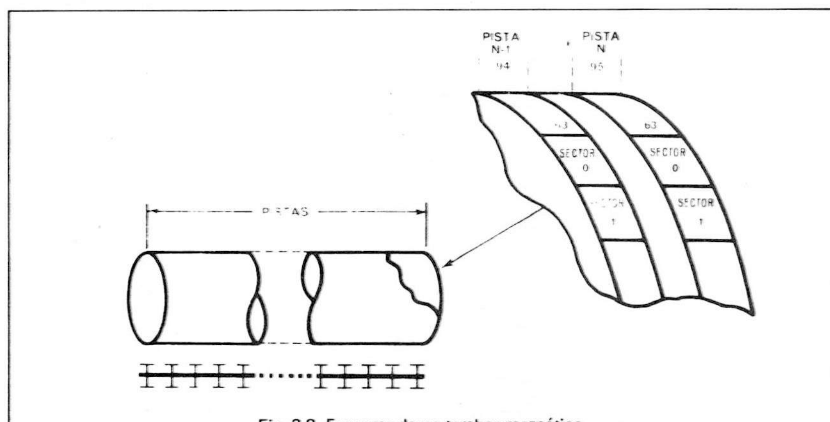



Fig. 3.8. Esquema de un tambor magnético.

Al existir una cabeza por pista se suprime el tiempo de posicionamiento que, recordemos, constituía el componente más elevado del tiempo total de acceso en los discos. En el caso de los tambores, el tiempo de acceso se compondrá únicamente del tiempo de selección de cabeza (despreciable) y del tiempo de espera de rotación, que será, por término medio, de una semirrotación. Normalmente, en los diferentes modelos de memorias de tambor, los tambores giran a una velocidad muy elevada, por lo que la espera de rotación suele ser menor que en las memorias de discos.

Algún modelo de tambor de gran capacidad tiene el juego de cabezas móvil, de forma que una misma cabeza tiene acceso a varias pistas consecutivas, lo que evidentemente redundará en perjuicio del tiempo de acceso.

Los tambores son normalmente direccionables por pista y sector, existiendo incluso tambores direccionables por pista, sector y palabra.

Los tambores, en comparación con los discos, tienen una capacidad y tiempo de acceso generalmente menores. Su precio, por bit almacenado, es, por el contrario, mayor.

Las características principales de los tambores más usuales oscilan entre los valores que a continuación se indican:

- Capacidad: 0,2 a 4 millones de caracteres.
- Tiempo medio de acceso: 3 a 10 milisegundos.
- Velocidad de transferencia: 200 a 1.500 K caracteres/segundo.

### 3.2.7. Diskette

El diskette o disco flexible ("floppy disk", en inglés), es un disco de material plástico fino recubierto de un material magnetizable, y de tamaño menor que los vistos anteriormente (aproximadamente de la dimensión de un microsurco de 45 r.p.m.). El floppy disk representa para los discos magnéticos lo que el cassette para las cintas, siendo, como éstos intercambiable. Su capacidad suele ser de 242 KB por cara.

Existen equipos llamados **grabadores de diskette** o **estaciones de diskette** que permiten realizar captura de datos sobre este tipo de soporte. Compuestos de un teclado y un elemento grabador, con una pantalla en la que aparecen los datos introducidos, así como mensajes de guía al operador, suelen disponer de cierto nivel de autonomía, con capacidad de almacenamiento, programación, funciones de control de datos, etc.

Los equipos más elaborados son conectables on-line a un ordenador central, pudiendo trabajar como **terminales inteligentes**. En este tipo de configuraciones lo normal es efectuar la conexión al final de la jornada, enviando por la línea a alta velocidad toda la información grabada a lo largo de dicho periodo.

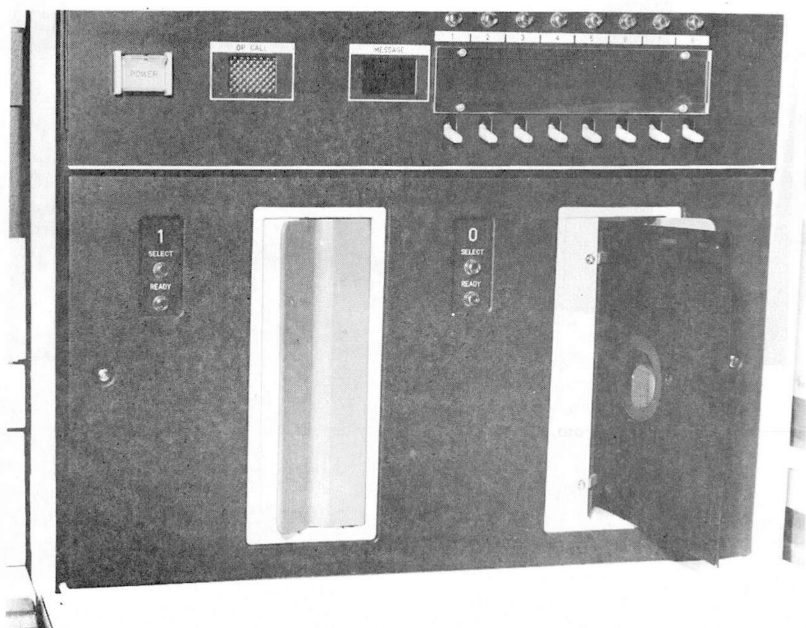



Foto 3.6. Unidad de diskette.

### 3.2.8. Lectores de marcas ópticas (tarjetas y documentos)

Las lectoras de tarjetas con marcas ópticas, de funcionamiento similar a las de tarjetas perforadas, reconocen la presencia o ausencia de una **marca oscura,** en vez de perforaciones. Por lo demás, las marcas de cada columna formarán determinado código.

Existen asimismo lectores de documentos (hojas de papel) con marcas ópticas. En uno y otro caso se detecta ópticamente la presencia de marcas oscuras en cierta posición. Los dispositivos de lectura permiten una velocidad de 100 tarjetas/minuto ó 30 documentos/minuto aproximadamente.

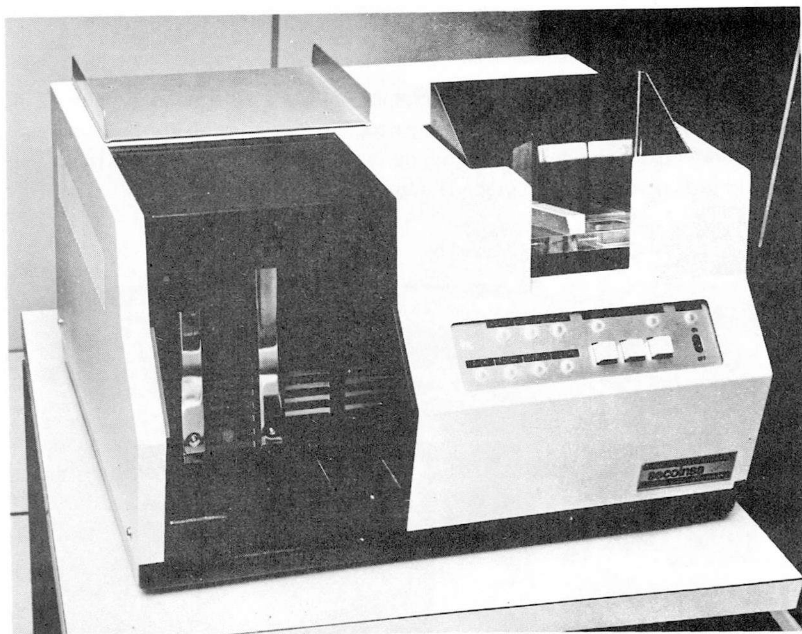


Foto. 3.7. Unidad de lectura de tarjetas marcadas.

### 3.2.9. Lectoras ópticas de caracteres

Los lectores de caracteres reconocen los símbolos impresos en cada documento dividiendo el carácter en su lectura en una matriz de puntos, cada uno de los cuales estará o no marcado. Una vez obtenida la **matriz de puntos,** se compara con los **caracteres-patrón,** hasta hallar uno con el que coincida.

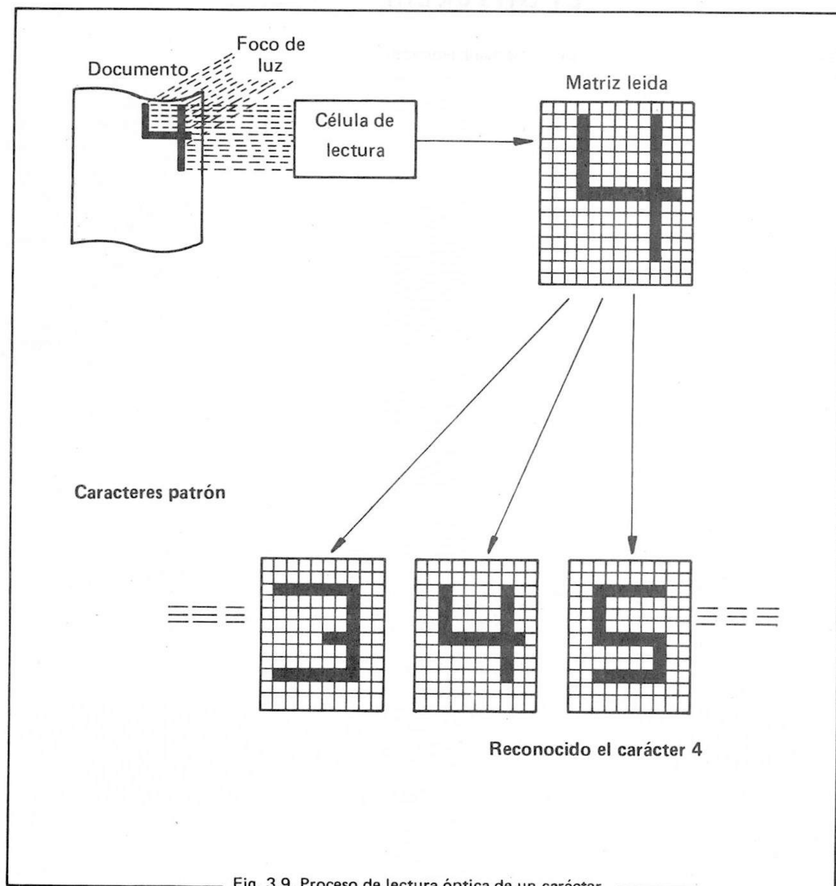



Fig. 3.9. Proceso de lectura óptica de un carácter.

Las lectoras pueden ser de caracteres **mecánicos** o **manuscritos, estilizados** o no, basándose todas en el mismo principio. Por otra parte, los caracteres pueden ser reconocidos **magnéticamente,** para lo cual se imprimirán con tintas magnetizables.

1234567890CNSTXZ⌿

1234567890

0 1 2 3 4 5 6 7 8 9

0123456789ЈΨнABCD
EFGHIJKLMNOPQRST...

Fig. 3.10. Caracteres estilizados.

### 3.2.10. Pantallas

Las unidades de pantalla están compuestas por una **pantalla** de rayos catódicos y un **teclado** numérico, alfanumérico y de funciones. Utilizando estos dispositivos, se puede establecer un proceso conversacional hombre-máquina, introduciendo datos al equipo por el teclado y obteniendo datos de salida por la pantalla. De esta forma, desde la pantalla pueden darse todo tipo de órdenes al sistema.

Las unidades de pantalla suelen utilizarse en sistemas **multipuesto,** en los cuales varios operadores comunican al tiempo con el ordenador. Su uso está muy extendido en aplicaciones bancarias, de reserva de plazas, etc, en los que junto a cada pantalla se suele instalar una impresora por la cual se edita la información que lo precisa.

Este tipo de periférico se utiliza tanto colocado cerca del ordenador como situado a distancia, como terminal remoto.

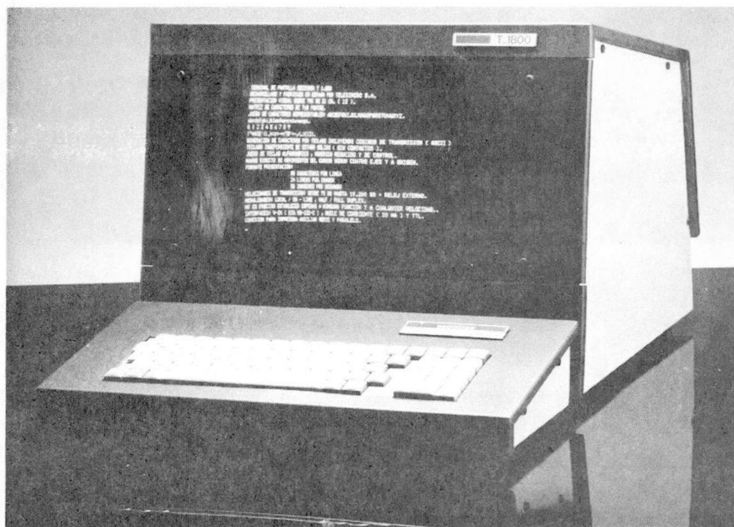


Foto 3.8. Unidad de pantalla.

### 3.2.11. Consolas de trabajo

Es otro tipo de periféricos utilizable en sistemas multipuesto. Consta de un teclado como el de las pantallas, o más complejo, una impresora serial, un pequeño visualizador de datos numéricos y puede llevar asimismo una lectora y/o perforadora de cinta de papel, así como un **insertor** de documentos.

De utilización cercana al ordenador, permite efectuar desde cada puesto una amplísima variedad de trabajos, tales como Contabilización, Facturación y otros parecidos, pudiendo procesar ciertos documentos en simultaneidad con el trabajo, por medio del insertor.



Foto 3.9. Consola de trabajo.

### 3.2.12. Consolas teletipo

Se utilizan para aplicaciones más sencillas que las de consolas de trabajo, y se componen de un teclado, una impresora serial y opcionalmente una lectora/perforadora de cinta de papel. Su campo de aplicación es el mismo ya descrito para los dos dispositivos anteriores.

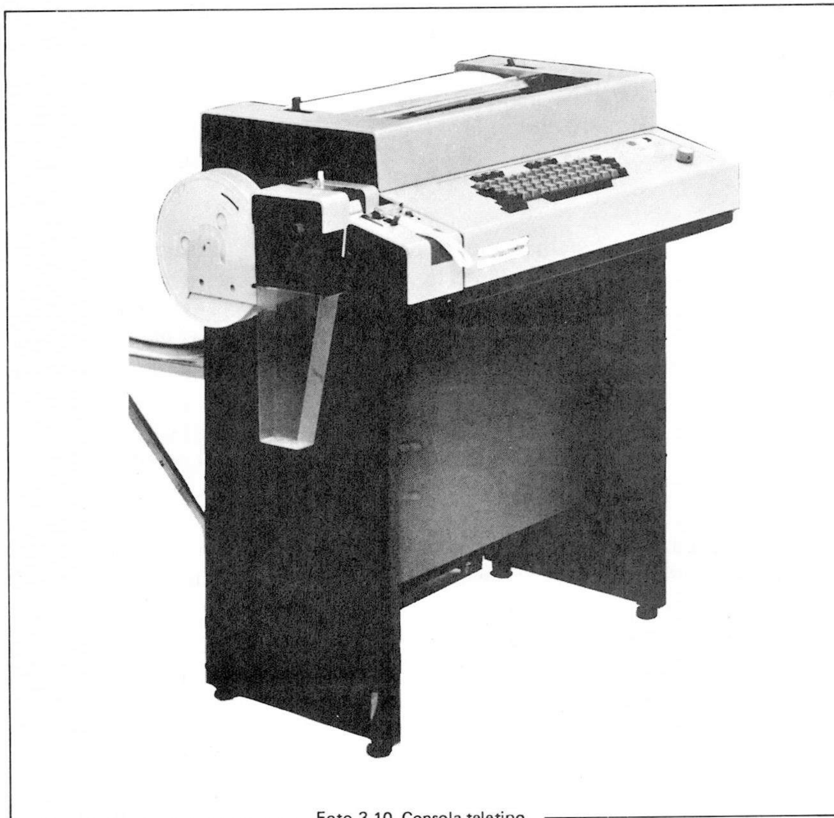


Foto 3.10. Consola teletipo.

### 3.2.13. Impresoras de caracteres

Un medio sencillo, y al mismo tiempo eficaz, para la comunicación de resultados obtenidos por un ordenador es el documento escrito. Este soporte de salida se puede conseguir mediante unas unidades denominadas impresoras.

Las impresoras de caracteres realizan la impresión carácter a carácter en forma secuencial, con un funcionamiento análogo al de una máquina de escribir, utilizando cinta entintada entre el papel y el dispositivo impresor. Para aumentar la velocidad de impresión las impresoras de caracteres suelen tener el carro fijo, siendo el **dispositivo portacaracteres** el que se desplaza a lo largo de la línea a imprimir. Este dispositivo puede adoptar formas muy variadas: barra, esfera, rueda, etc.

Las modernas impresoras tipo mosaico permiten alcanzar velocidades superiores trabajando igualmente bajo el principio de carácter a carácter. La impresión se realiza accionando contra el papel, con ayuda de electroimanes, unos punzones impresores, de entre los existentes en una **matriz de punzones,** componiendo de esta forma cualquier carácter a base de puntos de impresión tan próximos unos de otros que resulta difícil apreciarlos a simple vista.

Las velocidades que se consiguen con estas impresoras son del orden de 10 a 200 caracteres/segundo (según el modelo).

Algunas impresoras especiales aumentan su velocidad imprimiendo también de derecha a izquierda, por lo cual evitan el tiempo de posicionado después de la impresión de cada línea.

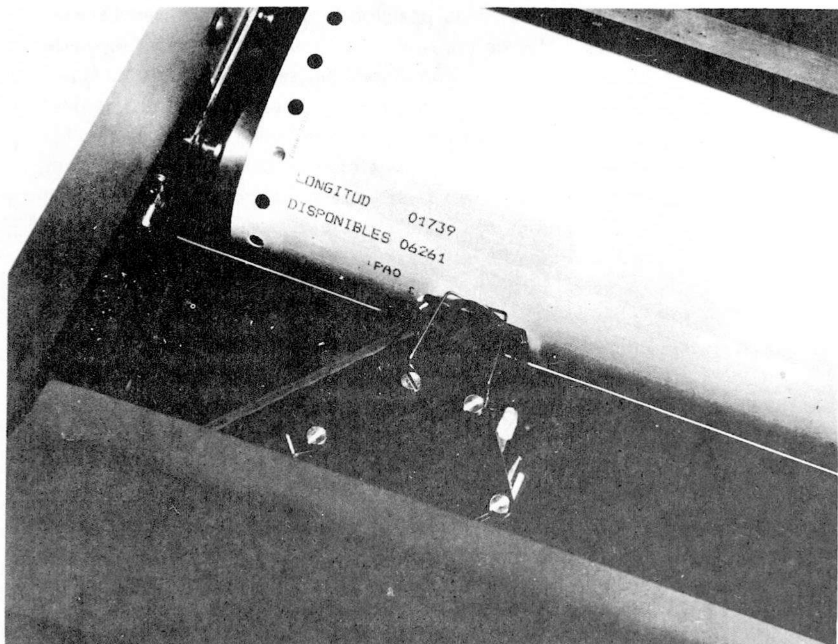



Foto 3.11. Impresoras de caracteres.

### 3.2.14. Impresoras de líneas

Las impresoras de líneas imprimen una línea prácticamente de un solo golpe, en lugar de hacerlo carácter a carácter. Esto implica que cada posición de la línea cuenta con un dispositivo que selecciona el carácter deseado. Basados en este principio existen impresoras que en cada posición tienen una **barra vertical** con todos los caracteres posibles; antes de imprimir una línea se desplaza el juego de barras, de forma que cada una de ellas sitúe el carácter seleccionado en cada posición, enfrente de un banco de martillos, uno para cada posición; así, al golpear simultáneamente los martillos contra las barras, se imprime la línea entera con los caracteres deseados. Una variante del procedimiento anterior consiste en disponer de un juego de **ruedas,** en lugar de las barras anteriores, una por posición, en cada una de las cuales están contenidos todos los caracteres posibles, seleccionando el carácter en cada posición mediante el giro que corresponda. Con cualquiera de estos procedimientos es difícil superar una velocidad de 500 líneas por minuto, ya que después de cada impresión el mecanismo de giro o desplazamiento se debe colocar en una posición origen, prefijada, para su posterior utilización.

Para eliminar estos inconvenientes se han desarrollado diversos tipos de impresora basadas en la impresión de los caracteres sin necesidad de detener el movimiento del dispositivo portacaracteres. Así, en una impresora parecida a la anteriormente descrita, un juego de ruedas, una por posición, conteniendo todos los caracteres que pueden imprimirse, gira continuamente delante de la línea a imprimir; cuando delante de cada posición el carácter a imprimir pasa ante el martillo, éste golpea el papel contra el carácter, produciéndose la impresión. Naturalmente, no se imprimen todos los caracteres de la línea en un solo momento, sino durante el corto intervalo de tiempo de la rotación completa de las ruedas. En la figura se representa el esquema de una impresora con nueve posiciones. Este mismo sistema se aplica en las llamadas impresoras **de tambor,** lo que equivale a haber hecho solidarias las ruedas del procedimiento anterior.

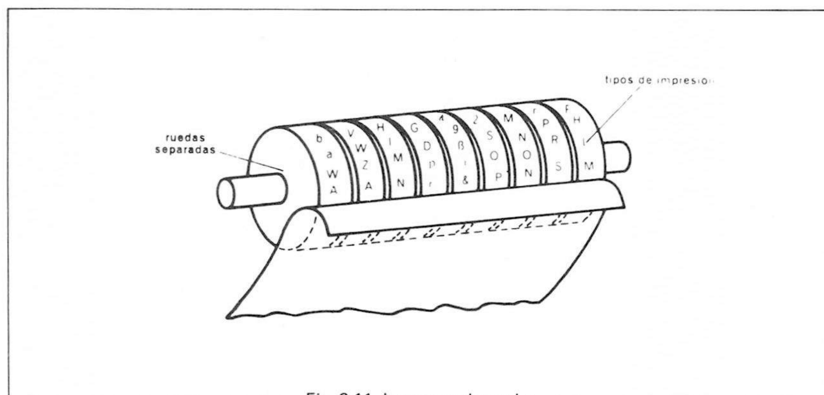



Fig. 3.11. Impresora de ruedas.

Para acelerar la velocidad de impresión en aquellos impresos en que muchas líneas contienen información exclusivamente numérica, se puede recurrir a

repetir dos veces los caracteres numéricos en las ruedas o circunferencia portacaracteres, en posiciones diametralmente opuestas. Con ello el tiempo medio de espera a que un carácter numérico dado pase por delante del martillo será de 1/4 de rotación en lugar de una semirrotación como sería el caso de un carácter no numérico.

En otro tipo de impresoras se emplea una **cadena** cerrada en forma de bucle que se mueve continuamente en un plano horizontal. La cadena contiene repetido varias veces (5 a 7 habitualmente), el juego completo de caracteres; cuando el carácter deseado pasa por la posición de impresión el martillo de esa posición es accionado magnéticamente contra la cadena.

Tanto en las impresoras de tambor como en las de cadena las posiciones de impresión por línea oscilan entre 120 y 160 y la velocidad de impresión entre 100 y 3.000 líneas por minuto.

En la figura se representa una impresora de cadena.

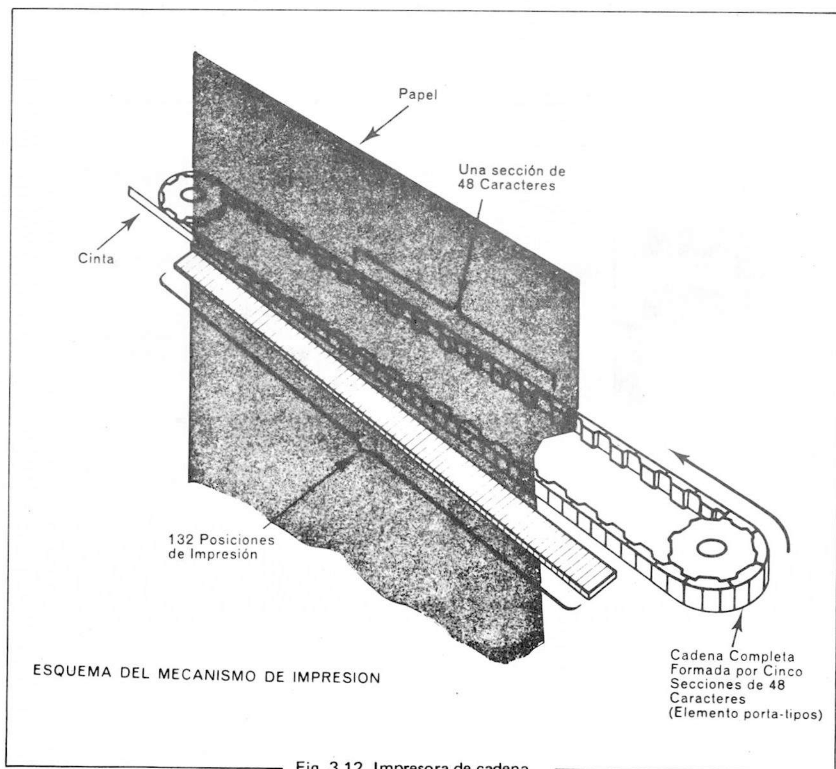



Fig. 3.12. Impresora de cadena.

Las impresoras empleadas actualmente trabajan con formularios de papel continuo que puede llevar incorporadas de 3 a 5 copias. El movimiento de formularios se gobierna mediante una cinta perforada. La cinta perforada de control, montada en forma de bucle, se mueve paralelamente al formulario de impresión, y tiene un número de canales de perforación del orden de 10 en una longitud que suele oscilar entre 18 y 44 cm.; longitudinalmente la cinta está dividida en secciones, de forma que una perforación se corresponda con un determinado renglón del formulario. Esta cinta gira, pasando delante de un sistema de fotocélulas que detecta las posiciones de perforación, mediante lo cual, y de acuerdo con el significado que se asigne a las mismas, se puede controlar el movimiento del papel.

Este mismo control se efectúa en otros casos por Software, llevando cuenta en memoria del número de líneas que se van imprimiendo.

Algunas impresoras trabajan con el sistema de **punzones** comentado en el apartado anterior. A base de un punzón por cada uno de los caracteres de la línea, se van golpeando o no los 35 puntos de cada carácter que componen la matriz (5 x 7), imprimiendo toda la línea en 35 pasos.

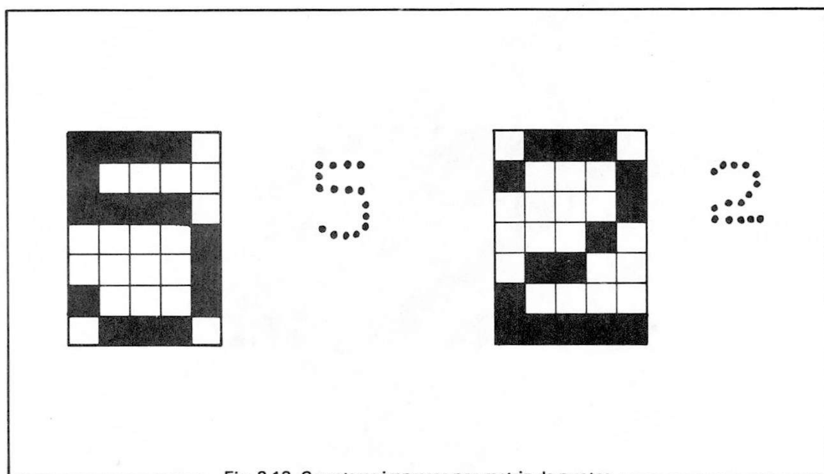


Fig. 3.13. Caracteres impresos por matriz de puntos.



Foto 3.12. Impresoras de líneas.

### 3.2.15. La consola del sistema

Un periférico que tienen todos los ordenadores, es la consola; este equipo cumple, como los anteriormente examinados, funciones de entrada y salida de información si bien estas funciones pueden considerarse distintas a las estudiadas en los anteriores periféricos, ya que consiste esencialmente en facilitar la comunicación entre el operador (persona encargada del control y manejo del ordenador) y el ordenador. A través de la consola el operador **recibe mensajes** del ordenador indicándole, por ejemplo, que ha finalizado un programa, la necesidad de cargar una unidad de cinta, etc.; por su parte el operador puede, a través de la consola, **dirigir órdenes** al ordenador, por ejemplo, comenzar la ejecución de un programa, interrumpir el desarrollo de otro, etc. Físicamente la consola puede estar constituida por un teclado y una impresora serial o bien un teclado y una pantalla.




Foto 3.13. Una unidad de pantalla como consola.

### 3.2.16. Fichas de banda magnética

Se conoce normalmente como ficha-cuenta magnética o como ficha-magnética simplemente. Su empleo se limita al ámbito de los ordenadores de gestión.

La ficha-cuenta magnética deriva directamente de la conocida **ficha contable** o ficha de posición, constituida por una cartulina con dos partes bien diferenciadas: una primera llamada cabecera, en la que figuran impresos una serie de datos invariables, y una segunda parte constituida por un número variable de renglones que han ido imprimiéndose sucesivamente en momentos diferentes, conteniendo en su conjunto una información de tipo histórico mas una información de situación, contenida esta última en el último renglón impreso.

Aclaremos esto con un ejemplo: supongamos un fichero de clientes constituido por un conjunto de fichas de cartulina de un tamaño cualquiera (una ficha por cliente). En la cabecera de la ficha figurarán impresos una serie de datos del cliente que serán invariables: nombre, domicilio, dirección, banco, crédito, etc. Cada operación que se realice con ese cliente se anotará en un renglón de la ficha, figurando el tipo de operación, valor de la misma, fecha y saldo final. El conjunto de renglones nos dará una información histórica sobre el cliente, mientras que el último renglón nos informará acerca de la situación actual o saldo de la cuenta del mismo.

La ficha de cuenta magnética añade a esta ficha contable un **soporte magnético** sobre el que se graban una serie de datos; fundamentalmente todos los de la cabecera y los correspondientes al renglón final. Se almacenan asimismo el número de renglón sobre el cual debe escribirse la próxima vez, lo que permite la alineación automática de la ficha cuenta magnética.

El soporte magnético de que consta la ficha puede ser de dos tipos:

a) Una **banda magnética** constituida por un material plástico recubierto por una capa de óxido ferromagnético, que se adhiere en uno de los bordes de la ficha, generalmente una banda en cada cara de la ficha. Esta banda es similar en su constitución a la empleada en los ordenadores electrónicos tradicionales.

b) Una capa de **tinta magnética** que se adosa a la ficha en posición similar a la anterior.

La información se graba en paralelo, sobre varias **pistas** longitudinales en que idealmente puede suponerse dividida la banda, existiendo, por tanto, una cabeza por pista.

La **capacidad de almacenamiento** es variable y depende, lógicamente, del tamaño de la cartulina. Con los formatos empleados usualmente varía entre 100 y 1.000 caracteres alfanuméricos por cara. El número de asientos o renglones a imprimir en cada caso suele variar entre 30 y 50 por cara.

El equipo lector de fichas-cuenta magnéticas suele constar de un dispositivo parecido a un buzón vertical donde se introduce la ficha que deseamos leer, y, generalmente, modificar. La ficha es arrastrada en forma que las cabezas de lectura de que está provisto el lector leen la información contenida en la banda, introduciéndose en la memoria central. Esta información permite en primer lugar alinear la ficha, situando el renglón a imprimir delante del dispositivo de impresión de que consta el lector (a veces el mismo de la impresora). Si deseamos modificar la información almacenada en la ficha se preparan estas modificaciones en la memoria central, grabándose a continuación sobre banda, con lo cual queda borrada la información anterior.

El operador deberá seleccionar manualmente la ficha-cuenta magnética que se desea leer o actualizar, de entre el conjunto de fichas-cuenta que constituye el fichero.

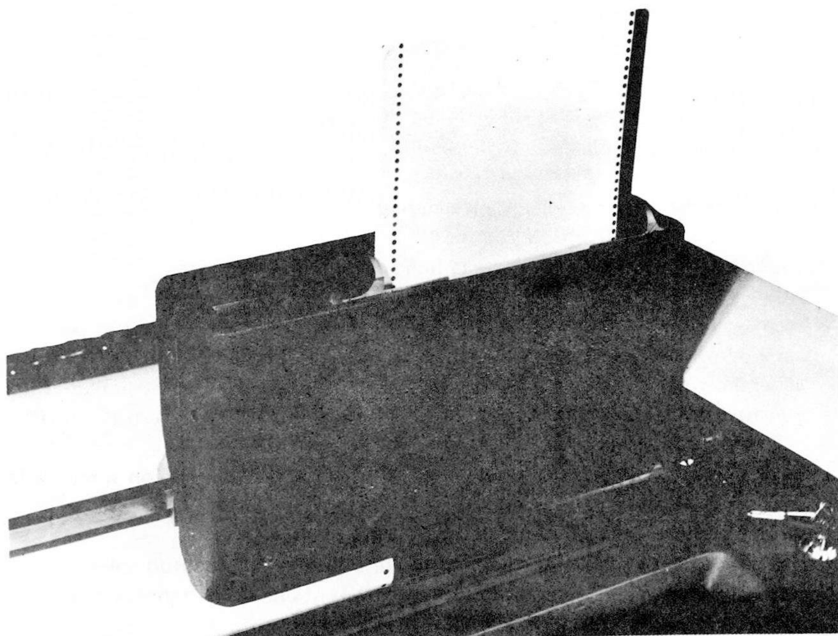


Foto 3.2. Lector-grabador de fichas de banda magnética.

### 3.2.17. Otros periféricos. Convertidores analógicos-digitales.

Los periféricos examinados hasta aquí son los más corrientemente utilizados. Para aplicaciones más especiales existen aún algunos otros. Así, podrían señalarse los **trazadores** o **plotter**, periféricos de salida capaces de, mediante el control del movimiento de una pluma sobre un papel, representar los resultados de salida de un ordenador en forma gráfica. Otros periféricos especiales proporcionan una salida **audible** que imita, con bastante exactitud, la voz humana.

Podemos señalar por último la importancia que tienen en aplicaciones de Control de Procesos los llamados **Convertidores Analógico–Digitales.** Su función es transformar señales continuas, como puede ser una curva de temperatura, presión, etc, en impulsos eléctricos inteligibles por el ordenador. Mediante estos elementos, es posible que el ordenador pueda recibir información de cualquier tipo de máquina, y por medio de convertidores digital-analógicos, manejar el ordenador aparatos de cualquier clase. De esta forma el ordenador no sólo podrá conocer datos analógicos cuantificados, sino manejar aparatos de control.

Un sistema de control de una fábrica puede efectuarse por medios totalmente automáticos, configurando lo que se conoce como **retroalimentación.** En la figura se muestra el esquema de un proceso mecanizado de tal forma: el ordenador recibe datos de presión, temperatura, velocidad del proceso, y si estos datos no se ajustan a los valores predeterminados, actúa para corregir el error abriendo válvulas, dando señales de aviso, reduciendo tensiones, etc. Al tiempo, puede ir almacenando los datos necesarios para elaborar posteriormente, o en el momento, informes sobre la marcha del proceso.

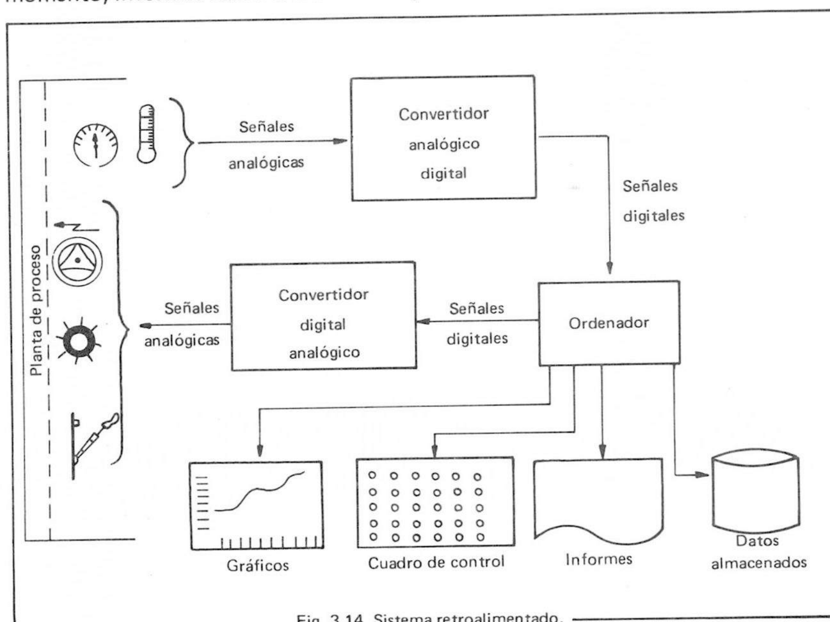



Fig. 3.14. Sistema retroalimentado.

Este tipo de sistemas controlan a los enfermos de un Hospital ingresados en la Unidad de Vigilancia Intensiva, los semáforos de una ciudad, el vuelo de un avión, la trayectoria de un cohete, etc.

## 3.3. Canales de Entrada/Salida

Vamos a estudiar a continuación cómo invierte un ordenador su tiempo de funcionamiento en ejecutar una tarea cualquiera; supóngase que esta tarea es la de leer un paquete de fichas perforadas, ejecutar una serie de procesos en la Unidad Central con los datos leídos en cada ficha y escribir unos resultados a través de una impresora de líneas.

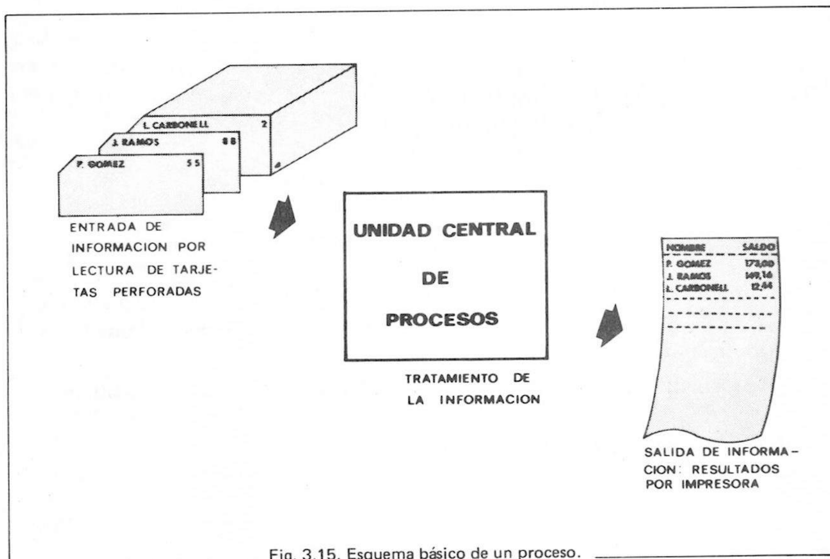



Fig. 3.15. Esquema básico de un proceso.

Si la lectura de las fichas se realiza a una velocidad de 600 tarjetas/minuto, el tiempo en leer una de ellas sería:

$$\frac{60 \text{ segundos}}{600} =$$

= 0,10 seg. = 100 milésimas de segundo.

Suponiendo que la impresora es capaz de imprimir 1.200 líneas/minuto, para imprimir los resultados en una línea tardará:

$$\frac{60 \text{ segundos}}{1.200} =$$

= 0,05 seg. = 50 milisegundos.

Suponiendo que el programa que procesa los datos de la ficha ejecutase un total de 200 instrucciones, y suponiendo que la Unidad Central invierte 20 $\mu$s. en ejecutar una instrucción (por término medio), el tiempo total que invertiría el ordenador en la ejecución del programa será:

200 instrucciones x 20 $\mu$s. = 4.000 $\mu$s. = 4 milisegundos.

En la siguiente tabla se resumen los tiempos anteriores:

| | En leer una ficha | En procesar una ficha | En escribir una línea |
|---|---|---|---|
| Tiempo total en efectuar la tarea por el ordenador ...................... | 100 ms. | 4 ms. | 50 ms. |

Resulta pues que el tiempo de ejecución del programa

$$\frac{4 \text{ ms.}}{154 \text{ ms.}} =$$

= 0,025, es decir, poco más del 2°/o del total.

En los ordenadores existentes hasta hace diez años aproximadamente, la Unidad Central permanecía totalmente inactiva durante el proceso de entrada/salida, esperando la culminación de esta operación para seguir ejecutando el programa, siendo pues su aprovechamiento extremadamente pequeño.

Los ordenadores actuales disponen de unos dispositivos llamados **canales** que unen la Unidad Central con los elementos periféricos permitiendo a aquélla continuar ejecutando instrucciones **mientras** se efectúan operaciones de entrada/salida, con lo que el rendimiento de los ordenadores es muy superior (por ejemplo, puede permitir la ejecución de varios programas "simultáneamente" en la Unidad Central; cuando un programa tiene que efectuar una operación de entrada/salida, otro de los programas almacenados en la memoria pasa a ejecutarse) mientras el canal, por sí solo, controla la operación de E/S. Este concepto se denomina **multiprogramación**.

Los canales pueden ser de entrada y/o salida y cada uno puede normalmente atender a diversos periféricos simultáneamente.

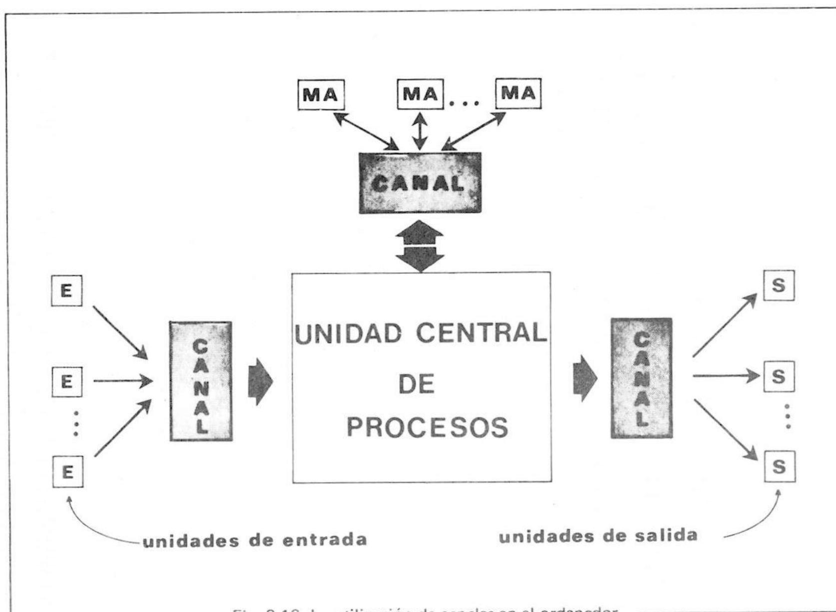



Fig. 3.16. La utilización de canales en el ordenador.

Pueden distinguirse en los canales dos formas diferentes de trabajo: **a ráfagas** o **multiplex**. Según la primera, el canal transmite hacia o desde la memoria central toda la información que lee o va a grabar en una unidad de entrada o salida, debiendo esperar las demás unidades de entrada/salida conectadas a él a que finalice la operación para poder comenzar a su vez una operación.

En la modalidad multiplex, el canal reparte su actividad en el tiempo, transmitiendo (en uno u otro sentido) cada vez un solo byte de/a cada unidad y pasando inmediatamente a la unidad siguiente. Así, los bytes pasan por el canal "entrelazados", según aparece esquematizado en la figura.

Las unidades más rápidas normalmente se conectan a canales que trabajan en la modalidad de ráfagas, mientras que las unidades más lentas –impresora, lectoras, perforadoras– se unen a canales que trabajan en modo multiplex.

Diversas firmas constructoras han popularizado el nombre de canales **selectores** para los canales que trabajan en modalidad de "ráfaga" y de canales **multiplexores** para los que lo hacen en modalidad multiplex.

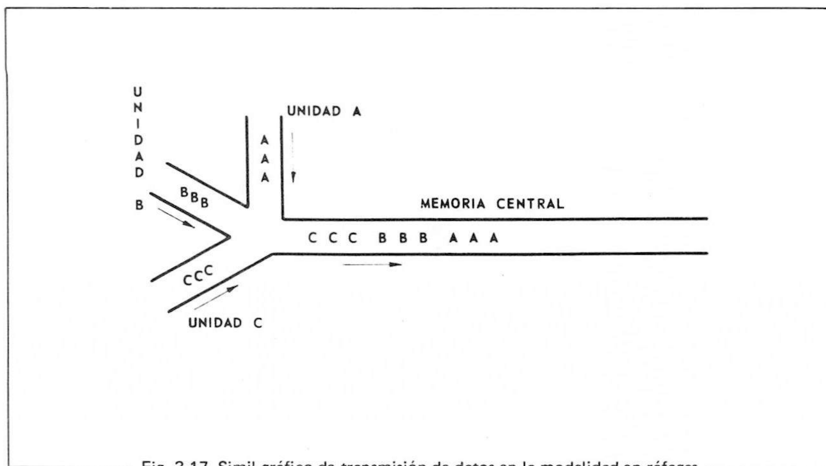



Fig. 3.17. Simil gráfico de transmisión de datos en la modalidad en ráfagas.

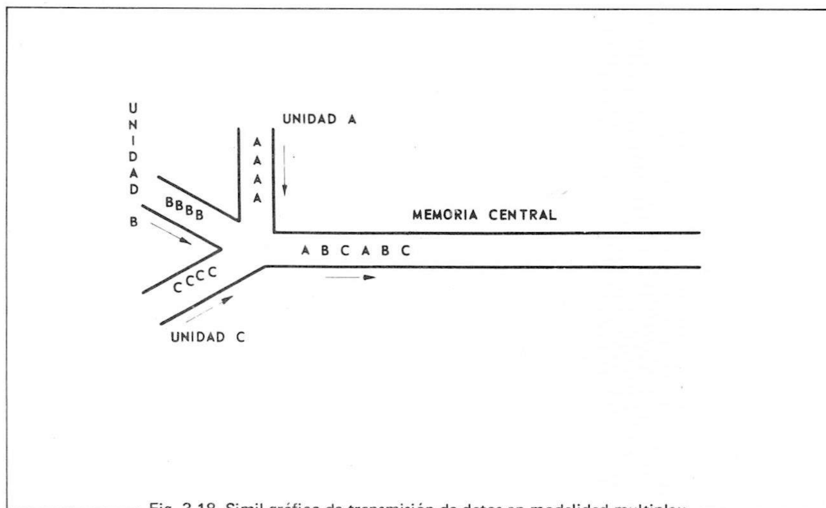



Fig. 3.18. Simil gráfico de transmisión de datos en modalidad multiplex.

Ultimamente se tiende a construir los ordenadores utilizando **Buses** a los que están conectados no sólo los periféricos sino también la CPU y la memoria central. Ello permite a los periféricos tener acceso directo a memoria (DMA) sin intervención de la CPU.

La utilización de dichos Buses posibilita al equipo incluso para conectar entre sí dos periféricos directamente.

La **anchura** de un Bus suele ser de una palabra, y por ellos se transfieren datos e instrucciones entre todos los elementos del ordenador:

CPU → Memoria
CPU → Periféricos
Periféricos → Memoria
Periféricos → Periféricos

Por este procedimiento, tanto la memoria como los periféricos se acceden por medio de **direcciones de memoria,** con lo que las instrucciones de referencia a la memoria pueden ser utilizados como instrucciones de E/S, lo cual proporciona una utilización muy efectiva del repertorio de instrucciones.

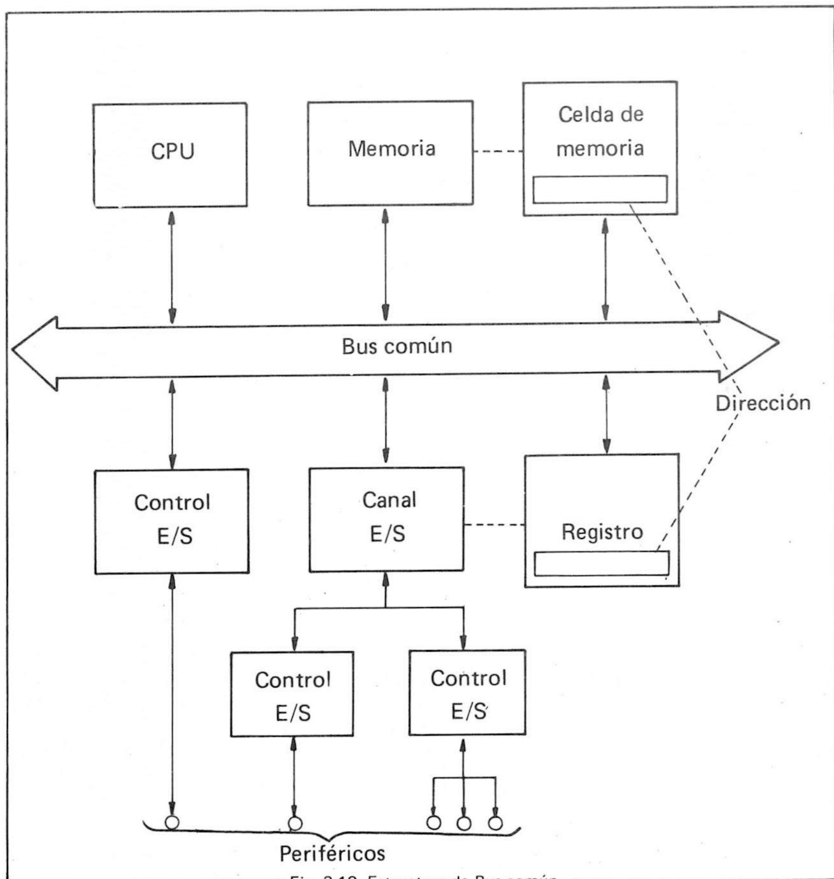



Fig. 3.19. Estructura de Bus común.

# CAPITULO 4. REPRESENTACION INTERNA DE LOS DATOS

## 4.1. Sistemas de numeración

Como ya se ha visto anteriormente, el ordenador maneja internamente datos binarios (cadenas de ceros y unos), por medio de los cuales representa números, nombres, etc. En este capítulo se verán los formatos internos de representación de los distintos tipos de datos, aunque en la mayoría de las ocasiones el programador trabaja en un **lenguaje de programación** lejano a la máquina y por tanto no necesita considerar el formato interno de los datos que maneja.

El hombre trabaja, normalmente, en sistema **decimal**, y el ordenador, en **binario.** Ambos son sistemas de numeración basados en los mismos principios. En ambos, la representación de un número se efectúa por medio de cadenas de símbolos, dependiendo el valor de cada símbolo de la posición que ocupe dentro de la cadena.

En el sistema decimal se utilizan diez símbolos (la base del sistema es 10) para representar valor numérico. Estos 10 símbolos son el 0, el 1, 2, 3, 4, 5, 6, 7, 8 y 9, por lo cual, para representar el valor **diez** no existe símbolo, utilizándose entonces para ello dos símbolos, un 1 y un 0 (10); al estar colocado en segunda posición, el 1 tiene un valor diez veces superior (Base = 10) al de los símbolos situados a su derecha. Un símbolo en tercera posición (una centena) valdrá diez veces más que una decena.

En el sistema binario la idea es idéntica. Se trabaja sólo con dos símbolos, el 0 y el 1. De tal forma, un valor 2 habrá que representarlo como 10, ya que el símbolo 2 no existe en binario. En este sistema, un símbolo en $2^a$ posición tendrá un valor dos veces superior (Base = 2) al de los símbolos en $1^a$ posición,

un símbolo en 3ª posición (lo que conocemos en decimal como centena), un valor dos veces superior al de aquellos en 2ª posición, etc.

Es decir, un número decimal $n_5n_4n_3n_2n_1$, donde n será cualquiera de los símbolos 0 a 9, valdrá:

$$n_5 \times 10^4 + n_4 \times 10^3 + n_3 \times 10^2 + n_2 \times 10^1 + n_1 \times 10^0$$

y en binario, un número igual, donde n serán símbolos 0 ó 1, valdrá:

$$n_5 \times 2^4 + n_4 \times 2^3 + n_3 \times 2^2 + n_2 \times 2^1 + n_1 \times 2^0$$

Así, por ejemplo, el valor del número en base decimal 1572 será:

$$1 \times 10^3 + 5 \times 10^2 + 7 \times 10^1 + 2 \times 10^0 = 1000 + 500 + 70 + 2$$

y el valor del número 10011101 en binario será:

$$1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$$
$$= 128 + 0 + 0 + 16 + 8 + 4 + 0 + 1 = 157$$

De la misma forma se puede trabajar con sistemas en cualquier base de numeración. Una bastante utilizada en Informática es la base 16 (sistema **hexadecimal**).

La razón de su uso es la siguiente: en un ordenador, al trabajar en sistema binario, un bit nos permite representar 2 valores, el 0 y el 1; dos bits permiten cuatro valores, 00, 01, 10 y 11; tres bits, 8 valores, 000, 001, 010, 011, 100, 101, 110 y 111. Para representar los diez símbolos del sistema decimal nos vemos obligados a utilizar cuatro bits. Pero cuatro bits permiten 16 valores, por lo cual trabajando en decimal desaprovechamos parte de su potencial de almacenamiento de información. Por ello, muchas veces los datos se almacenan en hexadecimal, y ocuparán menos que los mismos datos en decimal.

En hexadecimal trabajaremos con 16 **dígitos** (un dígito es cualquiera de los símbolos de un sistema de numeración). Como sólo tenemos diez dígitos numéricos, para representar los seis restantes se utilizan las letras A a F. Así, un valor diez en decimal será A en hexadecimal, un valor once, B, un valor quince se representará por F; para el valor dieciséis se utilizarán dos dígitos, 10, etc.

En general, dada una base B, el valor de un número $n_j\ n_{j-1}\ \ldots\ n_3\ n_2\ n_1$ expresado en dicha base será:

$$n_j \times B^{j-1} + n_{j-1} \times B^{j-2} + \ldots + n_3 \times B^2 + n_2 \times B^1 + n_1 \times B^0$$

Fig. 4.1.

Para cada uno de estos sistemas podemos construir tablas de operación:

| Tablas de Sumar | | |
|---|---|---|
| Hexadecimal | Decimal | Binario |
| 0 + 0 = 0<br>0 + 1 = 1<br>=<br>0 + 9 = 9<br>0 + A = A<br>=<br>1 + 8 = 9<br>1 + 9 = A<br>1 + A = B<br>=<br>5 + 9 = E<br>5 + A = F<br>5 + B = 0 (arrastre)<br>=<br>A + 3 = D<br>A + 4 = E<br>=<br>F + F = E (arrastre) | 0 + 0 = 0<br>0 + 1 = 1<br>=<br>0 + 9 = 9<br>1 + 0 = 1<br>1 + 1 = 2<br>=<br>4 + 5 = 9<br>4 + 6 = 0 (arrastre)<br>4 + 7 = 1 (arrastre)<br>=<br>9 + 7 = 6 (arrastre)<br>9 + 8 = 7 (arrastre)<br>9 + 9 = 8 (arrastre) | 0 + 0 = 0<br>0 + 1 = 1<br>1 + 0 = 1<br>1 + 1 = 0 (arrastre) |
| Tablas de multiplicar | | |
| Hexadecimal | Decimal | Binario |
| 0 x 0 = 0<br>0 x 1 = 0<br>=<br>0 x F = 0<br>1 x 1 = 1<br>= | 0 x 0 = 0<br>0 x 1 = 0<br>0 x 2 = 0<br>=<br>1 x 0 = 0<br>1 x 1 = 1 | 0 x 0 = 0<br>0 x 1 = 0<br>1 x 0 = 0<br>1 x 1 = 1 |

Fig. 4.2.

| Tablas de multiplicar (cont.) | | |
|---|---|---|
| Hexadecimal | Decimal | Binario |
| 1 x E = E<br>1 x F = F<br>=<br>5 x 9 = D (arrastra 2)<br>5 x A = 2 (arrastra 3)<br>5 x B = 7 (arrastra 3)<br>=<br>A x 9 = A (arrastra 5)<br>A x A = 4 (arrastra 6)<br>=<br>F x D = 3 (arrastra C)<br>F x E = 2 (arrastra D)<br>F x F = 1 (arrastra E) | =<br>4 x 5 = 0 (arrastra 2)<br>4 x 6 = 4 (arrastra 2)<br>4 x 7 = 8 (arrastra 2)<br>=<br>9 x 7 = 3 (arrastra 6)<br>9 x 8 = 2 (arrastra 7)<br>9 x 9 = 1 (arrastra 8) | |

Fig 4.2. (continuación)

| Ejemplo de Suma | | |
|---|---|---|
| Hexadecimal | Decimal | Binario |
| AD<br>+ 117<br>1C4 | 173<br>+ 279<br>452 | 10101101<br>+ 100010111<br>111000100 |
| Ejemplo de Multiplicación | | |
| Hexadecimal | Decimal | Binario |
| 2A<br>x 1A<br>1A4<br>2A<br>444 | 42<br>x26<br>252<br>84<br>1092 | 11010<br>x101010<br>00000<br>11010<br>00000<br>11010<br>00000<br>11010<br>10001000100 |

Fig. 4.3.

**Paso de Decimal a Binario y Hexadecimal**

Para pasar un número expresado en decimal al mismo número en otro sistema, se divide sucesivamente el número y los cocientes que van resultando por la base del sistema al que se quiere pasar. El resultado se obtiene por los restos que van quedando en las sucesivas divisiones.

| Paso del número 173 (en decimal) | |
|---|---|
| a Binario | a Hexadecimal |
| 173 ⌊ 2<br>13 86 ⌊ 2<br>1 06 43 ⌊ 2<br>0 03 21 ⌊ 2<br>1 01 10 ⌊ 2<br>1 0 5 ⌊ 2<br>1 2 ⌊ 2<br>0 1 | 173 ⌊ 16<br>013 10 |
| $173_{10} = 10101101_2$ | $173_{10} = AD_{16}$ |

Fig. 4.4.

**Paso de Binario y Hexadecimal a Decimal**

En general, el paso de un número expresado en base no decimal a base decimal se efectúa multiplicando cada dígito por su valor posicional, es decir, aplicando la fórmula expuesta en la figura 4.1.

| Paso al sistema decimal | |
|---|---|
| Del número AD en hexa | Del número 10101101 en binario |
| Valores hexa: | Valores binarios |
| --- \| 65.536 \| 4.096 \| 256 \| 16 \| 1 | – \| 128 \| 64 \| 32 \| 16 \| 8 \| 4 \| 2 \| 1 |
| $AD_{16} = A \times 16 + D \times 1 =$<br>$= 10 \times 16 + 13 \times 1 =$<br>$= 160 + 13 =$<br>$= 173_{10}$ | $10101101_2 = 1 \times 128 + 0 \times 64 +$<br>$+ 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 +$<br>$+ 0 \times 2 + 1 \times 1 =$<br>$= 128 + 32 + 8 + 4 + 1 = 173_{10}$ |

Fig. 4.5.

### 4.2. Datos numéricos

Para hablar de la representación interna de los datos numéricos nos apoyaremos en las ideas de bit y byte. Recordemos que un byte u octeto era el conjunto de 8 bits, 8 elementos de memoria que pueden tomar un valor 0 ó 1. Utilizaremos también la noción de **palabra de memoria.** Una palabra es la unidad elemental de tratamiento de la memoria, el conjunto de bits que el ordenador procesa en paralelo. El tamaño es variable de unos equipos a otros, y son normales las palabras de 16 y 32 bits.

Para operar con un número se debe conocer de él:

- Su signo.
- El valor de cada una de sus cifras, en su orden.
- El lugar de la coma decimal en la cadena de cifras.

Por ello, el ordenador deberá representar todo ello en memoria al almacenar un dato numérico. La forma de hacerlo varía de acuerdo al sistema de representación utilizado.

Los sistemas de representación se basan en dos tipos de notación:

- Coma fija.
- Coma flotante.

**a) Coma fija.**

El nombre viene de la forma de representar la posición de la coma decimal. El lugar viene predeterminado por el sistema, es decir, si el número va a ocupar 16 bytes, se asume que la coma estará implícita entre el 13º y el 14º, siendo los bytes 14, 15 y 16 los que almacenarán las cifras decimales. Existen tres formas de almacenamiento de números en coma fija:

- Binario puro.
- Decimal desempaquetado.
- Decimal empaquetado.

**a.1) Binario puro.**

En una palabra (o en una doble palabra si se requiere mayor precisión o capacidad) se almacena el valor binario del número.

De esta forma, si la palabra es de 4 octetos, dispondremos de 4 x 8 = 32 bits para representar un valor binario. Según esto, podremos almacenar un número de

$$2^{31} + 2^{30} + 2^{29} + \ldots + 2^3 + 2^2 + 2^1 + 2^0 = 2^{32} - 1$$

es decir, un número decimal de hasta 4.294.967.295.

En este sistema, los valores negativos se almacenan **complementados**, es decir, el resultado de restar de cero (en binario) el valor positivo.

**Ejemplo.–**

En una palabra de 2 bytes (16 bits) queremos almacenar el número 106. Se almacenaría:

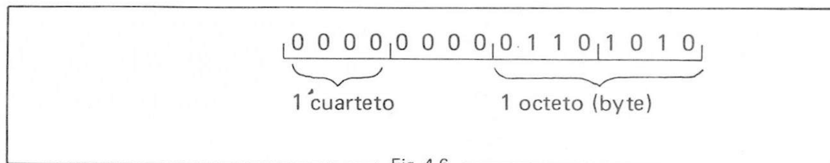



Fig. 4.6.

Si ahora quisiéramos representar el valor −106:

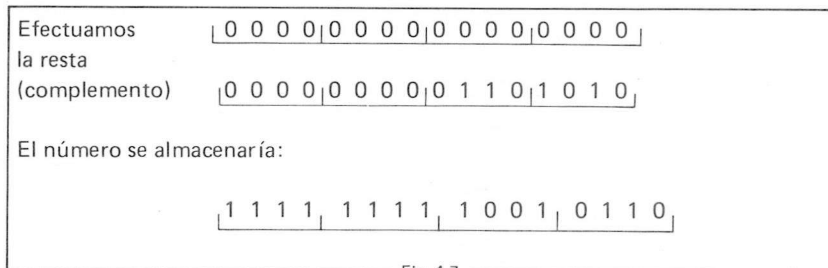



Fig. 4.7.

De esta forma, un número binario en positivo tiene ceros a la izquierda, y en negativo, unos, por lo cual para preguntar por el **signo**, el ordenador investigará el bit extremo izquierdo de la palabra en donde está almacenado el número.

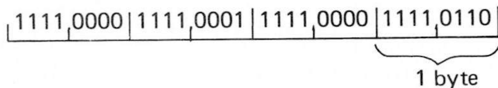
Este tipo de representación suele utilizarse para datos con los que se van a efectuar cálculos.

**a.2) Decimal desempaquetado.**

En este sistema, se almacena cada cifra del valor en un byte. Vimos que con 8 bits podíamos representar 256 valores, del 0 0 0 0 0 0 0 0 al 1 1 1 1 1 1 1 1

Dado que para representar las diez cifras del sistema decimal es suficiente con cuatro bits, el **cuarteto** izquierdo de cada byte estará lleno de unos.

El número 106 se almacenaría:




En notación hexadecimal sería:

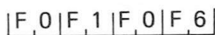
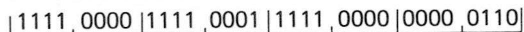



Fig. 4.8.

Para representar un valor negativo, sustituiríamos los unos del cuarteto izquierdo del byte extremo derecho por ceros. El número −106 quedaría:




En notación hexadecimal:

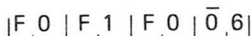



Fig. 4.9.

Esta representación suele utilizarse para Entrada/Salida de datos, ya que el Hardware de muchos periféricos trabaja con este tipo de representación.

**a.3.) Decimal empaquetado.**

Se utiliza para cálculo y almacenamiento de datos numéricos, y consiste en utilizar los cuatro bits de la izquierda de cada byte, desaprovechados en el sistema anterior, para almacenar otra cifra. Así, se almacenan dos cifras en cada byte. El número 106 quedaría:

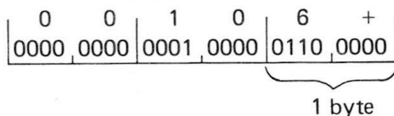



Fig. 4.10.

El signo ocupa el cuarteto derecho del byte extremo derecho, de tal manera que el número −106 se almacenaría:

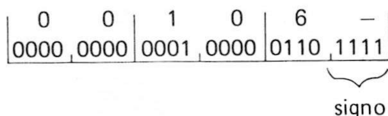



Fig. 4.11.

**b) Coma flotante.**

En coma flotante, para una base dada, los números se representan por medio de una **mantisa** y un **exponente,** como en la notación aritmética del mismo nombre. Así, el número 118,35 en coma fija, en coma flotante se representaría como $0,11835 \times 10^3$. De esta forma, el ordenador debe almacenar:

- El valor de la mantisa y su signo (+11835 en el ejemplo).
- El valor del exponente y su signo (+ 3 en el ejemplo).
- El valor de la base va implícito en el sistema elegido (10 en el ejemplo), puede ser 2, 8, 10, 16, etc.

El lugar de la coma decimal se supone a la izquierda de la mantisa. El signo será el de la mantisa.

La coma flotante puede presentarse en dos modos:

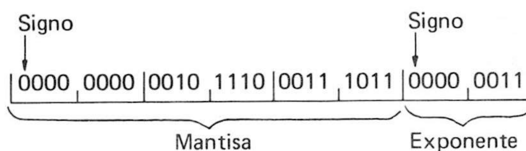
- Simple precisión.
- Doble precisión.

**b.1) Simple precisión.**

El número se almacena en una palabra, en binario puro, asignando un número de bits al exponente y otro a la mantisa.

**Ejemplo.**— El número 118,35 en palabra de 4 bytes.

$$118,35 = 0,11835 \times 10^3$$

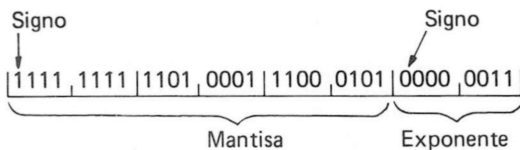



Mantisa = 11.835 (en 24 bits)
Exponente = 3 (en 8 bits)

Fig. 4.12.

El número –118,35 sería:

$$-118{,}35 = -0{,}11835 \times 10^3$$




Mantisa = –11.835
Exponente = 3

Fig. 4.13.

**b.2) Doble precisión**

Como el anterior, pero utilizando una **doble palabra** para almacenar el dato.

En el ejemplo, serían 8 bytes (64 bits) de los cuales se utilizarán 56 para representar la mantisa y 8 para signo y exponente.

Evidentemente, almacenando y trabajando con datos en doble precisión, es posible manejar números mucho mayores (o mucho menores) que en simple precisión. Este sistema suele utilizarse únicamente para aplicaciones muy especiales, en cálculos científicos en los que se requiere mucha precisión en los resultados o se precise trabajar con números muy altos o muy pequeños.

### 4.3. Datos alfabéticos y alfanuméricos

Este tipo de datos se almacena siempre en formato desempaquetado, es decir, un carácter por byte, ya que en este caso sí son necesarios más de cuatro bits para representar cada carácter. La configuración de ceros y unos que represente a cada carácter dependerá del **código** utilizados por el ordenador. En equipos con celdas de 8 bits (bytes) los más utilizados son los códigos EBCDIC y ASCII.

Representación en código ASCII (Paridad par por el 8º bit)

| S | E | C | O | I | N |
|---|---|---|---|---|---|
| 1100 1010 | 1010 0011 | 1100 0011 | 1111 0011 | 1001 0011 | 0111 0010 |

Fig. 4.14.

## 4.4. Matrices y tablas

Cuando en una aplicación informática se deben manejar un cierto número de datos que tienen las mismas propiedades, por ejemplo, los diez posibles descuentos a aplicar a una venta, dependiendo del volumen de compra, puede resultar muy conveniente procesarlos en formato de **tabla.** Aunque estos datos pueden describirse individualmente, en la práctica puede ser mucho más sencilla la elección de uno de ellos por medio de **subíndices** de valor variable en ejecución.

Al definir una tabla se asigna el mismo nombre a todos los elementos que la componen, diferenciando unos de otros por medio de una clave para cada elemento o bien por medio de subíndices. Así, si a los descuentos antedichos se les asigna el nombre DESTO, el $1^{er}$ descuento sería DESTO (1), el $2^{o}$, DESTO (2), hasta DESTO (10), que sería el último. Esto sería una **matriz unidimensional,** una tabla.

Siguiendo con el ejemplo, podemos suponer que existen 30 descuentos, diez para los clientes tipo 1, según el volumen de compra, otros diez para los de tipo 2 y otros diez para los de tipo 3. En ese caso formaríamos una matriz **bidimensional** de 3 x 10 elementos. En ella, DESTO (1,1) sería el descuento $1^{o}$ aplicable a los clientes tipo 1, DESTO (2,7) sería el $7^{o}$ descuento para clientes tipo 2, etc.

Lógicamente no habría limitación en el número de dimensiones de una matriz, si bien en la práctica los compiladores suelen limitarlo a uno, dos o tres.

La figura 15 muestra la estructura lógica de matrices de una, dos y tres dimensiones. Un número superior no es representable gráficamente.

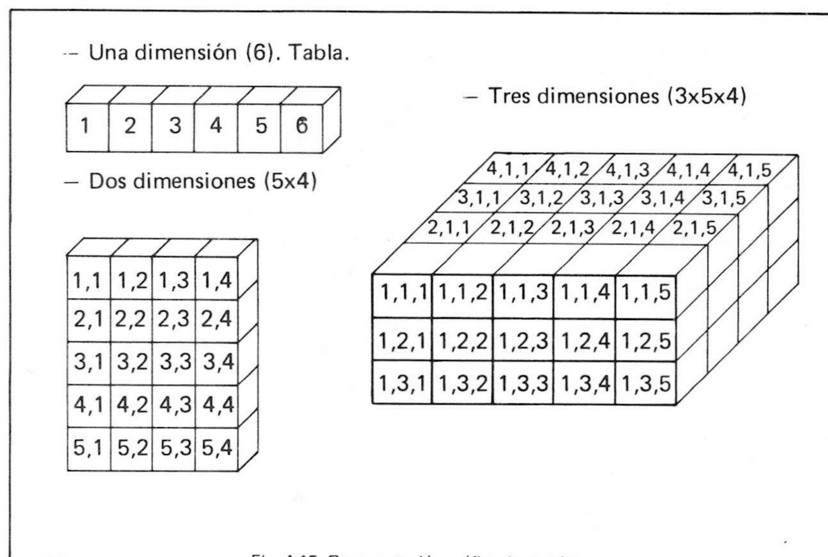



Fig. 4.15. Representación gráfica de matrices.

La representación en memoria de una matriz se efectúa a base de ubicar los elementos uno detras de otro, es decir, una **cadena de elementos** del tamaño definido El número de elementos será el resultado de multiplicar los valores máximos de cada dimensión de la matriz. Así, una matriz tridimensional de 3 x 5 x 4 tendrá 3 x 5 x 4 = 60 elementos.

Cuando un programa quiera acceder a un elemento específico de una matriz, el ordenador deberá calcular el sitio exacto dentro de la memoria donde deberá estar dicho elemento, o bien se deberá acceder por programa a cada uno de los elementos hasta identificar el especificado por posición o clave.

En una matriz definida A(I, J, K), el elemento A(i, j, k) se hallará en:

Dirección = (i x (J x K) + j x (K) + k − 1) x L + B

L = Tamaño de cada elemento
B = Base (Dirección en memoria del 1$^{er}$ elemento de la matriz)

Fig. 4.16.

**Localización en una matriz de cualquier tamaño**

Matriz M $(D_1, D_2, D_3 \ldots D_n)$
Elemento M $(d_1, d_2, d_3 \ldots d_n)$

Dirección = $(d_1 \times A_1 + d_2 \times A_2 + \ldots + d_{n-1} \times A_{n-1} + d_n - 1) \times L + B$

Aj = Amplitud de la dimensión j

$$Aj = D_{j+1} \times D_{j+2} \times \ldots \times D_{n-1} \times D_n$$

Fig. 4.17.

# CAPITULO 5. LENGUAJES DE PROGRAMACION

## 5.1. Introducción

Como ya se dijo anteriormente, el objetivo de la programación es la codificación de **algoritmos,** listas de instrucciones que especifican una secuencia de operaciones que resuelven un problema determinado.

Un algoritmo sirve para resolver un determinado problema para cualquier posible conjunto de valores de sus variables: un programa trabajará igual sean cuales sean sus datos de entrada.

Un **programa** es un conjunto de sentencias que forman la representación, inteligible por el ordenador, de un algoritmo.

Una **sentencia** o instrucción es una cadena de símbolos de cierto alfabeto. Esta cadena se formará de acuerdo a ciertas reglas sintácticas, y construida de tal forma tendrá un cierto sentido (es la semántica de la frase).

Un lenguaje de programación posee y queda definido por un **alfabeto,** ciertas reglas de **sintaxis** y una **semántica.** Sucede igual que en los lenguajes naturales, si bien en éstos últimos la sintaxis y semántica son tan complejas que ninguno ha podido ser aún definido totalmente.

En los primeros ordenadores, para introducir un programa en la memoria era necesario algún procedimiento material para "grabar" un conjunto de unos y ceros en cada celda, hasta completar el número total de instrucciones de que consta el programa.

Antes de la introducción del programa había que "escribirlo" en el lenguaje que entiende el ordenador (que en el futuro se denominará **lenguaje máquina**) es decir, a base de ceros y unos sobre un documento o papel. A continuación, a la vista de esta información escrita, introducir de alguna forma, (por ejemplo utilizando tarjetas o cinta de papel previamente perforada con ceros y unos), cada instrucción en la celda de memoria respectiva hasta completar la "carga" del programa.

Semejante trabajo era sumamente tedioso, sometido a un elevado porcentaje de error y requería una gran cantidad de tiempo (considérese que un programa normal puede tener varios miles de instrucciones). (Ver figura 1).

Por los motivos enunciados anteriormente, fue necesario desarrollar algunas "herramientas" que permitiesen aliviar la escritura de los programas y su posterior transcripción a la memoria central en forma de instrucciones máquina.

Estas herramientas son los llamados **lenguajes de programación.** Estos permiten escribir las instrucciones del programa, no a base de ceros y unos, sino mediante símbolos adecuados, (letras, signos numéricos y especiales) es decir, con caracteres fácilmente inteligibles. A estos tipos de lenguaje se les denomina **simbólicos,** y es lo que se conoce normalmente como lenguaje de programación.

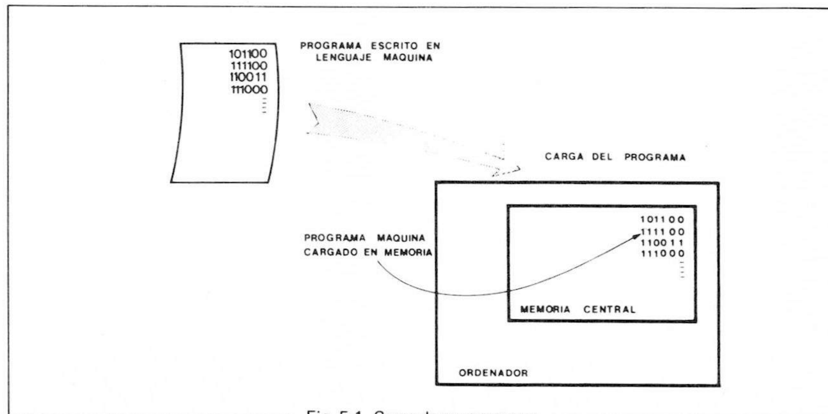



Fig. 5.1. Carga de un programa.

## 5.2. Tipos de lenguajes de programación

Según se ha visto, la primera clasificación que se puede hacer con los lenguajes, consiste en dividirlos en:

- Lenguajes de máquina.
- Lenguajes simbólicos.

Los primeros ya son conocidos y son los que están escritos directamente como sucesión de unos y ceros. Cada ordenador tiene su propio lenguaje máquina que dispone de su juego o repertorio particular de instrucciones.

Los lenguajes simbólicos pueden clasificarse a su vez en:

A) Lenguajes orientados hacia la máquina, o lenguajes de bajo nivel.
B) Lenguajes orientados hacia el hombre, o lenguajes de alto nivel.

### A) Lenguajes de bajo nivel.

Las instrucciones escritas en estos lenguajes, guardan una cierta analogía con las instrucciones de lenguaje de máquina en que, posteriormente, serán traducidas. (Puede, por ejemplo, distinguirse perfectamente si el ordenador para el que se escribe el programa tiene lógica de 1, 2 ó 3 direcciones).

Las principales características de las instrucciones escritas en estos lenguajes son las siguientes:

a) El código de operación es de tipo **nemotécnico,** lo que facilita su aplicación por el programador. Así por ejemplo, una instrucción de sumar tendrá un código de operación parecido a SUM (sumar) o ADD (adición).

b) Las **direcciones** de los operandos o datos, que en el lenguaje máquina eran numéricas absolutas, pasan a ser **simbólicas,** es decir, se le atribuye un **nombre** a cada dato y la instrucción hace referencia a dicho nombre. Así, por ejemplo, una instrucción para sumar dos cantidades que se podrían llamar CANT 1 y CANT 2, situadas respectivamente en las celdas 10.125 y 10.280 se escribiría como sigue

   - Lenguaje de máquina: 17 10.125 10.280 (todo en binario).
   - Lenguaje simbólico: SUM CANT 1 CANT 2.

c) Cada instrucción escrita en lenguaje simbólico, genera, al traducirse, **una** sola instrucción en lenguaje máquina (de ahí que a este tipo de lenguaje suele conocerse con la denominación de "uno a uno"). Existen, sin embargo, lenguajes orientados a la máquina que utilizan algunas instrucciones, conocidas como **macro-instrucciones**, que al traducirse producen **varias** instrucciones máquina.

Los lenguajes de este tipo suelen denominarse lenguajes **Assembler (o ensamblador)** conociéndose con igual denominación a los programas **traductores** correspondientes, lo que puede plantear cierta confusión al lector poco iniciado. Así por ejemplo, se dice que un programa escrito en assembler es traducido por un traductor assembler.

De forma general se puede decir que todos los suministradores proporcionan los ordenadores acompañados, como mínimo, de un programa traductor tipo ensamblador que permite programar el equipo en el lenguaje correspondiente. Debido a que los diversos modelos tienen su propio lenguaje máquina, los lenguajes tipo assembler son diferentes para cada ordenador.

**B) Lenguajes de alto nivel (o evolucionados)**

Las instrucciones de que consta un programa fuente escrito en un lenguaje de este tipo no recuerdan en absoluto el "hardware" del ordenador ni su funcionamiento interno como ocurría con los lenguajes de bajo nivel; no tienen pues ningún parecido o analogía con las instrucciones de lenguaje máquina. Estos lenguajes permiten escribir instrucciones orientadas al problema que se quiere resolver y no al ordenador que va a ejecutar el programa correspondiente. Por tal motivo dichos lenguajes emplean terminología fácilmente comprensible y que se aproxima más o menos al propio lenguaje humano.

Cada instrucción de programa escrito en lenguaje evolucionado se traduce en general a varias instrucciones máquina, al contrario de las instrucciones escritas en lenguaje de bajo nivel; (se expresa gráficamente lo anterior diciendo que los lenguajes de alto nivel permiten escribir instrucciones muy **potentes**, ya que una sola instrucción puede producir varias de lenguaje máquina).

El desarrollo de estos lenguajes se ha debido, lógicamente, al intento de facilitar una programación rápida y su empleo, efectivamente, no requiere en absoluto del conocimiento de la estructura de los ordenadores, lo que permite al programador de aplicaciones dirigir sus esfuerzos directamente al problema que pretende resolver.

Existe otra diferencia sustancial entre estos lenguajes y los orientados a la máquina. Así como estos últimos proliferan paralelamente al número de modelos diferentes de ordenadores, dado que cada ordenador dispone del suyo propio, no ocurre lo mismo con los lenguajes de alto nivel o evolucionados que, al no estar orientado el ordenador, pueden ser **universales**. Esto quiere decir que los lenguajes evolucionados más comunes pueden utilizarse para todos los ordenadores siempre que éstos tengan el traductor correspondiente. La compatibilidad de estos lenguajes es **casi** general en todos los ordenadores (existen excepciones en que, aún siendo básicamente iguales los lenguajes empleados, difieren ligeramente de unos a otros ordenadores).

Para que se advierta la potencia de un,lenguaje de alto nivel, la figura 2 presenta un ejemplo de las instrucciones precisas para resolver un problema matemático en lenguaje FORTRAN y, comparativamente, las instrucciones que hacen falta en un lenguaje tipo assembler.

Las instrucciones del lenguaje assembler empleadas, se han elegido de dos direcciones y son:

MOV: mover un dato de una posición a otra (o trasladarlo).
SUM: sumar dos datos.
MLT: multiplicar dos datos.
DIV: dividir dos datos.

Obsérvese de paso los códigos nemotécnicos empleados en este último, según se explicó anteriormente.

El problema a resolver es calcular el valor de una función X de la expresión matemática

$$X = \frac{(A + B) . (C + D)}{E + F}$$

A, B, C, D, E y F se interpretan como variables. Se entiende que el valor de estas variables debe conservarse en todo momento.

**(a) En lenguaje de bajo nivel (tipo "Assembler").**

MOV B, M: Llevar valor de B a M.
SUM A, M: Sumar A y M. El resultado dejarlo en M.
MOV D, N: Llevar valor de D a N.
SUM C, N: Sumar C y N. El resultado dejarlo en N,
MOV F, P: Llevar valor F a P.
SUM E, P: Sumar E a P. El resultado dejarlo en P.
MLT M, N: Multiplicar valores M y N. El resultado dejarlo en N.
DIV N,P: Dividir valor N entre valor P (es decir dividir (A + B) . (C + D) entre E + F). El resultado dejarlo en P.
MOV P, X: Llevar P (el resultado) a X.

**(b) En lenguaje evolucionado (FORTRAN):**

x = (A + B) * (C + D)/(E + F)

Fig. 5.2.

Ventajas e inconvenientes de la utilización de los distintos lenguajes

Es suficiente referirse a las ventajas e inconvenientes de los lenguajes de alto nivel para deducir las correspondientes a la utilización de los de bajo nivel.

**—Las ventajas de la utilización de lenguajes de alto nivel**

Son múltiples las ventajas que representa la utilización de estos lenguajes, entre las que se pueden destacar:

- **Facilidad de su aprendizaje**, dada la simplicidad de sus instrucciones y la "inteligibilidad" de las sentencias empleadas.
- **Sencillez de utilización** dado que el usuario, al escribir sus programas, no tiene por qué tener presente el funcionamiento interno del ordenador que va a utilizar, puesto que estos lenguajes están orientados al problema a resolver y no a la máquina que lo resuelve. No solamente es una ayuda importante durante la escritura de los programas, sino en la fase de prueba de los mismos.
- **Potencia de las instrucciones.** Ya se indicó anteriormente que una sentencia de programa escrito en lenguaje evolucionado se traducía, tras la compilación, en varias instrucciones del programa objeto. Esto representaba una enorme ventaja al programador, al poder condensar en pocas sentencias multitud de operaciones que requerirían un número importante de instrucciones en un lenguaje poco evolucionado.

**— Los inconvenientes de la utilización de lenguajes de alto nivel**

Existen también inconvenientes en la utilización de estos lenguajes, entre los que se pueden destacar los siguientes:

- La utilización de estos lenguajes puede obligar a utilizar unos elementos de hardware en el ordenador, adicionales o ampliados; como ejemplo, es necesaria en general más memoria central, debido a que los programas compiladores ocupan bastante memoria, dada la complejidad de sus funciones; la utilización de los compiladores implica además el uso de periferia auxiliar que quizás no hubiese hecho falta en otro caso. Por ello los ordenadores más pequeños casi nunca incluyen traductores de lenguaje de alto nivel (pequeños ordenadores de gestión, miniordenadores, etc.).

- Los programas objeto, producidos en la traducción efectuada por el programa compilador, son generalmente menos eficientes en ocupación de memoria y en tiempo de ejecución (ocupan más memoria y tardan más tiempo en ejecutarse), que los que se hubiesen conseguido mediante el empleo de lenguajes de bajo nivel, ya que en éstos el programador "sigue más de cerca" el trabajo de la máquina y puede aprovechar mejor todos sus recursos (*)
- Un coste adicional al sistema debido a la complejidad que lleva consigo el diseño de los traductores de lenguajes, por las casas constructoras.

---

*Suele emplearse una analogía entre un ordenador utilizando lenguaje evolucionado o no, y una persona que intenta amueblar su casa con muebles terminados standard, o bien muebles a medida. Las ventajas de la rapidez de instalación, precio y comodidad de adquisición en el primer caso, lleva como contrapartida la poca o inadecuada utilización del espacio de la casa.

### Conclusiones más importantes

Es un hecho evidente que, cada vez más, se tiende a utilizar lenguajes de programación de alto nivel; las razones que se pueden aducir pueden ser las siguientes:

- Los costes de personal han crecido, (lógicamente lo continuarán haciendo), de forma ininterrumpida, por lo que los usuarios se ven forzados a buscar sistemas que reduzcan al máximo el trabajo de personal que dichos ordenadores requieren. No cabe duda que la utilización de los lenguajes de alto nivel contribuye a un mayor ahorro en la programación de los sistemas.
- Los costes del hardware de los ordenadores han disminuido comparativamente dentro de los costes globales de los sistemas informáticos. Es un hecho observable que cada vez más, equipos u ordenadores calificados tradicionalmente como pequeños, de acuerdo con su precio, incluyen un hardware suficientemente adecuado para permitir el uso de lenguajes evolucionados.

Puede matizarse todo lo anteriormente expuesto con los siguientes puntos:

a) **Según la frecuencia de uso de la aplicación.** En principio los lenguajes de alto nivel deberán utilizarse más en aquellos programas o aplicaciones que se ejecuten pocas veces en el futuro. Es preferible sin embargo, la utilización del lenguaje de bajo nivel cuando los programas a desarrollar se vayan a ejecutar muy frecuentemente, puesto que este tipo de programas son muy eficientes y aprovechan más óptimamente las posibilidades del sistema. Estas ideas quedan plasmadas en la figura 3. En ésta, se representan unas curvas de costes de la ejecución de un programa en función del tiempo o frecuencia de utilización. En la curva (1) se representan el coste de una aplicación desarrollada en un lenguaje de bajo nivel. A tiempo 0 de explotación (coste de desarrollo de la aplicación), el coste es alto debido a la complejidad que representa programar en dicha lenguaje; sin embargo, la pendiente de la curva en la explotación es más baja que la de la curva (2), correspondiente a una aplicación desarrollada con un lenguaje de alto nivel. En esta última curva se

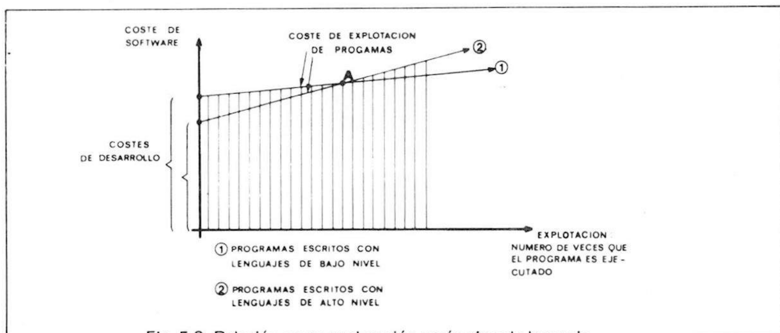


Fig. 5.3. Relación coste explotación según tipo de lenguaje.

observa un coste de desarrollo más bajo debido a la facilidad de empleo de este lenguaje.

El punto A viene a representar el umbral donde empieza teóricamente a ser beneficioso el empleo de lenguajes de bajo nivel, dada la frecuencia de utilización del programa.

b) **Según el tamaño de los ordenadores.** En los grandes ordenadores, se emplean casi siempre lenguajes de alto nivel, puesto que importa relativamente menos que los programas objeto producidos ocupen más memoria.

c) **Aplicaciones de complejidad especial.** En algunas aplicaciones especiales existen fuertes condicionantes en cuanto a tiempo de ejecución, (ordenadores al servicio de la reserva de plazas de ferrocarril o líneas aéreas, ordenadores trabajando en consulta rápida de cuentas corrientes, o bien aquellos aplicados al control de naves espaciales). La programación para este tipo de aplicaciones, por exigir unas condiciones especiales de eficacia y rapidez, parece en principio más adecuada para ser efectuada en lenguajes de bajo nivel.

### 5.3. Los traductores de lenguajes

En el apartado anterior se vió como los programas se escriben en lenguajes evolucionados más cercanos al hombre que a la máquina.

Sin embargo, el ordenador no es capaz de utilizar un programa escrito en caracteres de este tipo. Por lo tanto, el programa escrito de esta forma en hojas de papel, hay que traducirlo al lenguaje que entiende el ordenador, es decir, al lenguaje máquina.

En principio, el traductor podría ser una persona que mediante un "diccionario" (en que apareciera la sucesión de unos y ceros que correspondería a cada símbolo del programa escrito en lenguaje simbólico), fuese introduciendo por algún procedimiento la sucesión de bits que correspondería en cada celda de memoria, hasta terminar de cargar el programa en lenguaje máquina completo (figura 4).

Dado el poder de cálculo y procesamiento que tiene un ordenador, son los propios equipos los que pueden efectuar perfectamente esta tarea de traducción de lenguajes. Recordando que un ordenador ejecuta un programa utilizando unos datos de entrada y obtiene un resultado, se han confeccionado programas llamados **traductores** que, utilizando como datos de entrada las instrucciones de un programa escritas en caracteres inteligibles o simbólicos (también llamado **programa fuente**), dan como resultado el programa escrito en lenguaje máquina que le corresponde (también llamado **programa objeto**). (Ver figura 5).

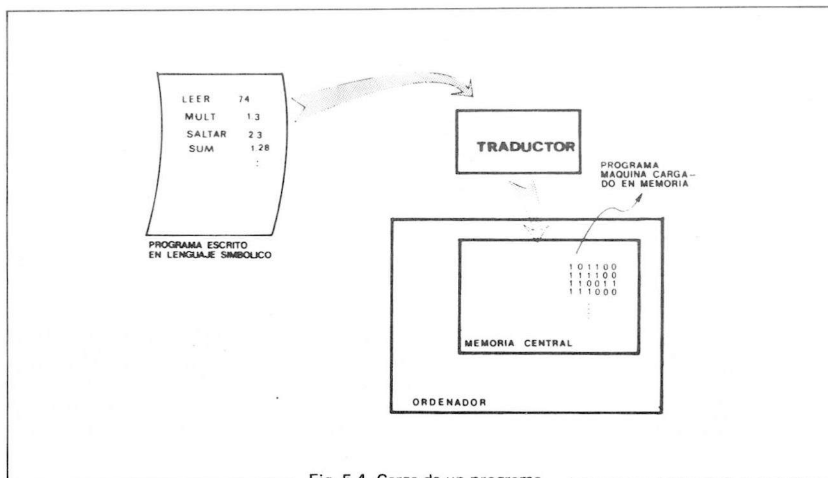
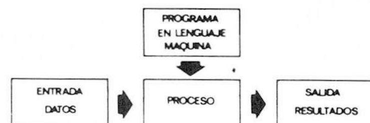



Fig. 5.4. Carga de un programa.




(A) ESQUEMA DE UN PROGRAMA EN LENGUAJE MAQUINA (EL PROCESO ES DIRECTO)




(B) EJECUCION DE UN PROGRAMA EN LENGUAJE SIMBOLI-CO PREVIAMENTE HAY QUE TRADUCIR EL PROGRAMA FUENTE

Fig. 5.5. Ejecución de programas.

Normalmente el proceso real consiste en escribir el programa fuente en unas hojas especiales, llamadas **hojas de codificación**.

Dichas hojas son fácilmente transcribibles por ejemplo, a tarjetas perforadas, de manera que cada fila de la hoja de codificación, que normalmente contiene una instrucción en lenguaje simbólico, es transcrita sobre una tarjeta perforada (es decir, cada instrucción fuente se codifica en una tarjeta). El lote de tarjetas perforadas (programa fuente), es leído por una unidad de entrada, en este caso por un lector de tarjetas del ordenador, el cual mediante la ejecución del programa traductor, obtiene como resultado de su trabajo el programa traducido o programa objeto. Normalmente dicho programa obtenido no queda definitivamente en memoria central, sino que se transfiere a un soporte de memoria auxiliar (cinta o disco magnético, etc.) para su posterior utilización.

De esta manera el proceso de traducción es ejecutado una sola vez, y el programa objeto obtenido se puede guardar indefinidamente para ejecutarlo cuantas veces se requiera, introduciéndole cada vez en memoria desde el soporte auxiliar donde esté almacenado.

Al ejecutarse la traducción de un programa fuente, el traductor se encarga de "revisar" si las instrucciones del programa, escritas en lenguaje simbólico, obedecen a las reglas del lenguaje que utiliza. En caso de encontrar incorrecciones de símbolos o de sintaxis, da aviso de las mismas mediante textos sobre una impresora y no procede a la traducción propiamente dicha hasta que el programador corrija adecuadamente las instrucciones erróneas. Una vez que el traductor ha verificado la corrección de todas las instrucciones que componen el programa fuente, se efectúa la traducción.

Normalmente proporciona simultáneamente a través de una impresora un listado del programa fuente con su traducción o programa objeto paralelamente, indicando la posición de memoria que ocupará cada instrucción máquina, número de orden de cada instrucción del programa, memoria total que ocupa el programa, etc.

Este listado es sumamente importante en la fase de depuración y prueba del programa para el correcto seguimiento del mismo, y en su caso, para poder efectuar modificaciones. Los procesos explicados se esquematizan en la figura 6.

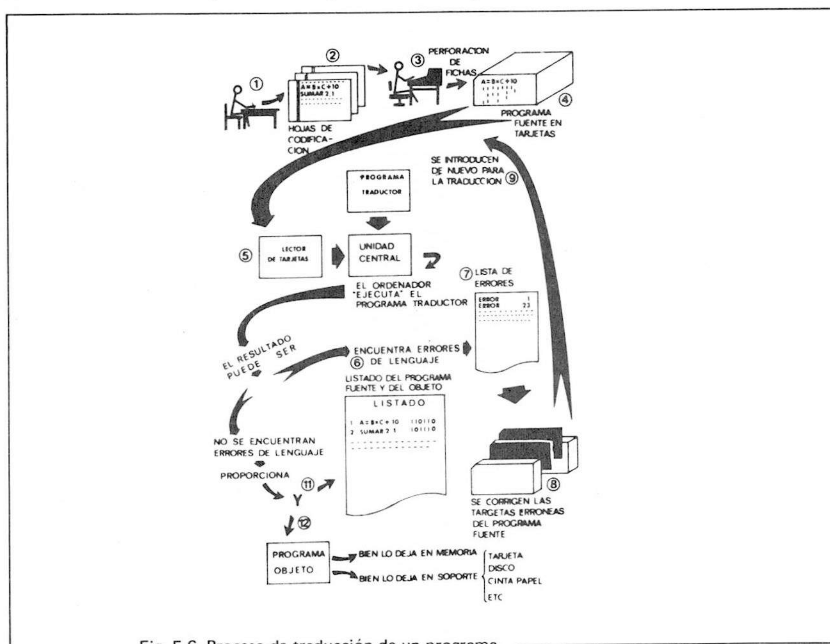



Fig. 5.6. Proceso de traducción de un programa.

Podemos señalar por último los tipos de traductores, de acuerdo a su forma de traducir:

– **Ensambladores**:
  – La traducción se hace de uno a uno. Una instrucción del programa fuente da

lugar a una instrucción en lenguaje máquina. Son excepciones las macroinstrucciones.
- Se emplean para traducir programas escritos en lenguaje ensamblador. Son lenguajes orientados a la máquina.

- **Compiladores**:
  - La traducción se hace de una a varias. Una instrucción del programa origina varias en lenguaje máquina.
  - Se emplean para traducir programas escritos en lenguajes orientados al hombre, lenguajes de alto nivel.

- **Intérpretes**:
  - Es un caso especial en que cada instrucción se ejecuta tras ser traducida.
  - En las versiones de intérpretes puros o de semi-intérpretes (hay una traducción parcial previa), se emplean para facilitar la modificación de programas fuente, en simulaciones y en lenguajes de control.

# CAPITULO 6. REGISTROS Y FICHEROS

## 6.1. Definición de fichero, registro lógico y registro físico

Los ordenadores trabajan no sólo con datos individuales, como puedan ser un precio o un código, sino con conjuntos de datos lógicamente relacionados. Por ejemplo, todos los datos de un cliente formarían el **registro** de ese cliente, y varios registros de clientes distintos configurarían el **fichero** de clientes.

En las aplicaciones mecanizadas, el ordenador deberá efectuar operaciones con grandes conjuntos de datos, que para poder ser manejados racionalmente, poseerán cierta estructura lógica, y estarán almacenados de acuerdo a otra estructura, estructura física.

Estos grandes conjuntos de datos reciben el nombre de ficheros.

La noción de fichero viene de aquellos que se emplean en cualquier oficina, en la que un conjunto de fichas de cartulina contienen la información de los clientes, de los productos del almacén, de las cuentas contables, etc.

En los primeros sistemas de proceso de datos se mantenían las fichas (ahora perforadas) para almacenar esta información. La idea seguía siendo guardar en cada ficha la información sobre un cliente, un artículo, etc.

Con la aparición de los modernos soportes de información (cinta magnética, tambores, discos, etc.) comienza a verse la necesidad de aprovechar más racionalmente los soportes. Así, si una cinta graba bloques de 1024 bytes y el tamaño del **registro** de un cliente (la información que se desea guardar de cada cliente) son 200 bytes, parece lógico agrupar los clientes en grupos de cinco, en vez de desa-

provechar 824 bytes de cada bloque. De la misma forma puede ocurrir que se necesiten 2000 bytes por cliente, debiendo grabar entonces dos bloques por registro.

Lo anterior nos introduce en los conceptos de **registro lógico**, que suele conocerse simplemente como registro, y **registro físico**. Un registro lógico es un conjunto coherente de informaciones agrupadas (el conjunto de los datos de un cliente, de un artículo, etc.); un registro físico es la unidad de información que un determinado soporte lee o graba de una sola vez, y será un bloque de cinta, un sector de disco, etc. El tamaño de un registro físico depende de cada dispositivo, y no tiene por qué guardar relación alguna con el tamaño de los registros lógicos. Es muy frecuente llamar **bloque** al registro físico.

Según lo dicho, en un bloque podrá haber un registro, varios, o sólo parte de uno.

Normalmente los lenguajes de programación de alto nivel poseen instrucciones de lectura y grabación de registros, encargándose el compilador de que el programa objeto genere las necesarias lecturas o grabaciones de bloques. Ello libera al programador de la necesidad de tener en cuenta, al nivel de codificación de instrucciones, la diferencia entre registro y bloque en cada dispositivo.

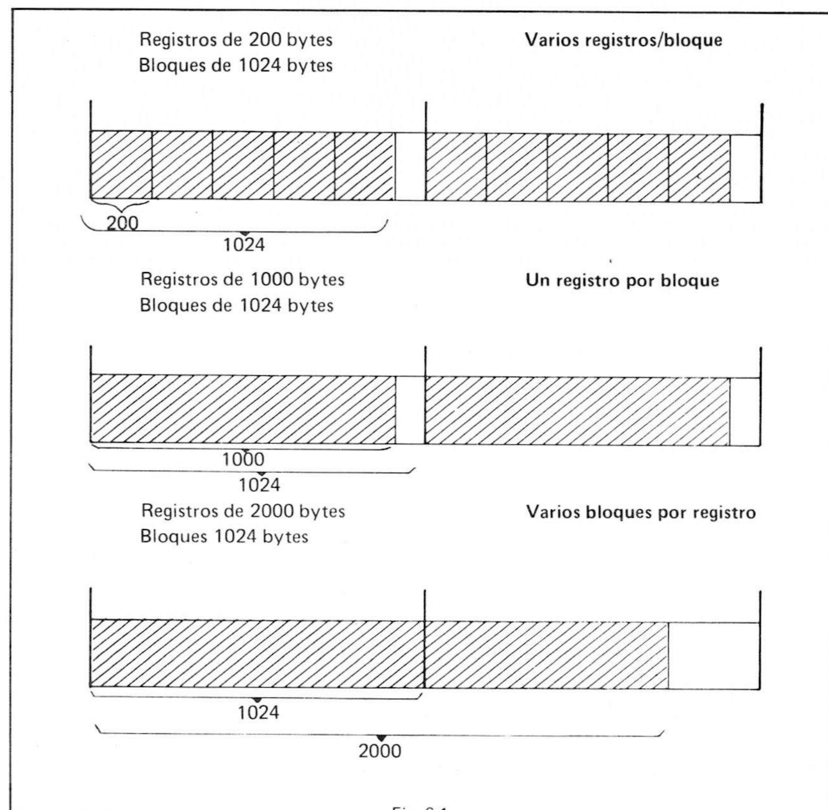



Fig. 6.1.

## 6.2. Tipos de ficheros según su función

Según el objeto a que se destinan, los ficheros pueden agruparse según el siguiente cuadro:

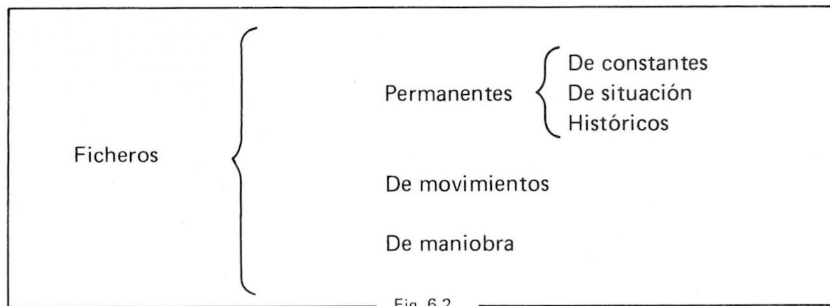



Fig. 6.2.

a) **Ficheros permanentes.** Son todos aquellos cuya vida no se extingue con un tratamiento o proceso, sino que permanecen para ser tratados y consultados cualquier número de veces. El fichero de clientes de una oficina es un ejemplo de este tipo.

Un fichero **de constantes** es aquél cuya información no se altera nunca o casi nunca, por la naturaleza de los datos que contiene. Un ejemplo de ello sería un fichero en el que figurasen los valores de una tabla de logaritmos, invariable, y otro, el fichero de personal de una empresa, muy poco variable.

Se llama fichero **de situación**, y más corrientemente **fichero Maestro** a aquél que en cada momento contiene la información actualizada. Así, ficheros que contengan el stock de un almacén, o el estado de las cuentas de clientes, deben estar por su naturaleza siempre al día. Cualquier variación es registrada en ellos con la máxima urgencia.

Los ficheros **históricos** son los que contienen información acerca de las situaciones ya pasadas. Un ejemplo sería un fichero que contuviese los resultados de los balances mensuales de una empresa. Esta información no está al día, sino que es resultado de tratamientos anteriores y suele utilizarse para establecer estadísticas.

b) **Ficheros de movimiento.** También se conocen como ficheros de transacciones, y contienen los datos necesarios para actualizar o consultar un fichero permanente. Un fichero que contuviese las salidas de almacén ocurridas en el día sería el fichero de movimientos con cuyos datos se actualizaría el fichero maestro de almacén. Una vez efectuado el proceso, el fichero de movimientos pierde su validez.

c) **Ficheros de maniobra.** O ficheros de trabajo, o temporales, son aquellos que el ordenador crea con los resultados intermedios de un proceso, para ser utilizados inmediatamente por el mismo proceso. Su validez es la del proceso de que se trate.

## 6.3. Registros

Dentro de un registro estarán contenidas diversas informaciones, cada una de las cuales será un **campo** del registro. Cada uno de estos campos puede estar dividido a su vez en otros **subcampos**, o grupo de datos relativos a un concepto; por ejemplo en un registro de un cliente, **Nombre, Dirección, Saldo,** serían campos del registro, mientras que **Domicilio** y **Localidad** podrían ser los dos subcampos que conforman el campo **Dirección**.

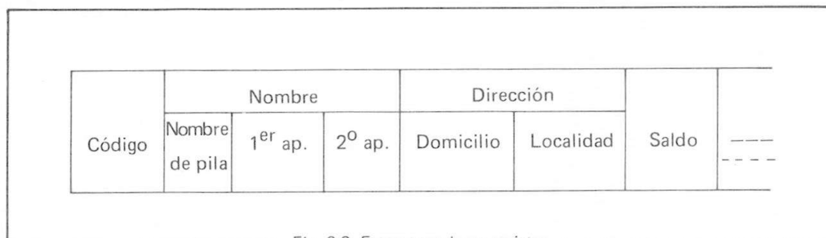



Fig. 6.3. Estructura de un registro.

En un registro puede haber uno o varios **indicativos**. Indicativo es aquel campo que permite localizar dentro del fichero un determinado registro. Así, si queremos buscar el cliente cuyo código es 1137, el campo **código** será el indicativo utilizado para buscar dicho registro. Igualmente se podría haber utilizado el campo **Nombre** como indicativo pues también serviría para **identificar** un cliente.

Al diseñar un registro se pueden plantear problemas para definir su longitud, por varias causas.

- **Campos de longitud variable.** Por ejemplo, la localidad. En ciertos casos será Cádiz y en otros Puerto de Santa María. Se puede tomar la máxima longitud posible como tamaño del campo, en cuyo caso se desaprovecha mucho espacio. En el caso de información que lo permita, se pueden buscar abreviaturas y tomar un tamaño menor al máximo posible.

- **Campos con número variable de subcampos.** En un fichero de clientes en que se guarde la cifra de ventas por producto, determinado cliente puede comprar sólo de un artículo, o hacerlo de todos los posibles. Como en el caso anterior, si se tomase el número máximo posible se desaprovecha espacio.

- **Campos no existentes en todos los registros.** Por ejemplo, puede ser necesario prever en el fichero de clientes una dirección para envío de la mercancía y otra para envío de las facturas, caso que se sabe sólo se dará en muy pocos clientes. Ello desaprovecha igualmente el soporte.

Según esto, diseñar registros de longitud fija origina grandes pérdidas de espacio en el soporte. Pese a ello es lo más usual en programas de aplicación, ya que el

manejo de registros de longitud variable presenta gran dificultad en programación, a no ser que el Sistema Operativo o el lenguaje de programación prevean dicha posibilidad. En el caso de adoptar registros de longitud variable, a fin de que el programa, al leer un registro, sepa qué longitud tiene cada campo y qué campos lo integran, se utilizan diversos métodos:

— Colocar delante de cada campo variable su longitud.
— Utilización de símbolos especiales de fin de campo y fin de registro.
— Incluir un campo en primera posición que indique la configuración y/o tamaño del registro. Puede ser un simple código o una **máscara** con cierta complejidad.

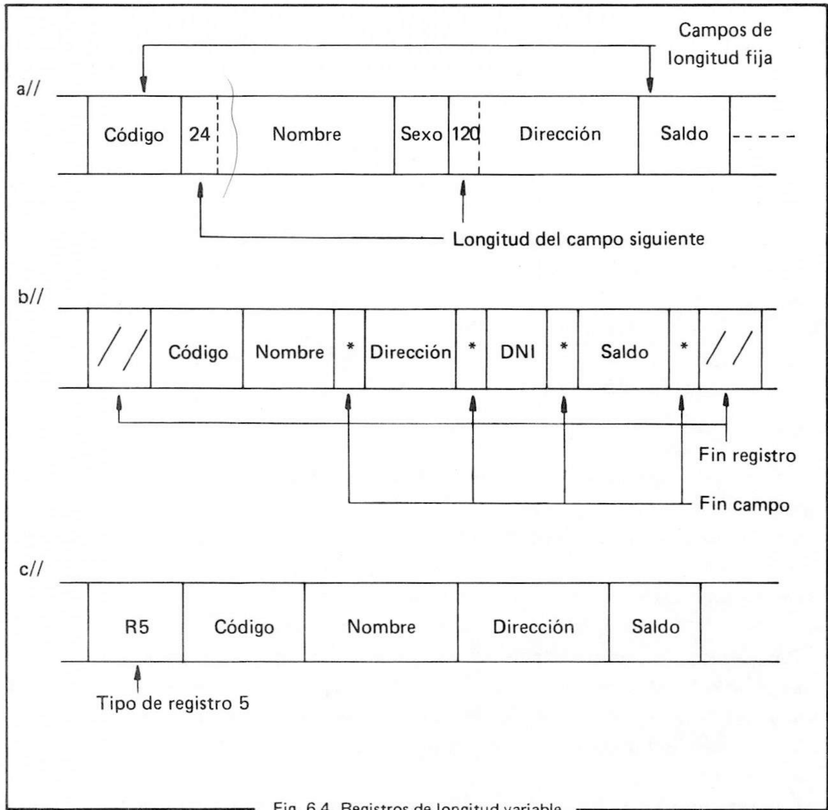


Fig. 6.4. Registros de longitud variable.

## 6.4. Ficheros ordenados

En muchos casos es necesario que un fichero esté ordenado de acuerdo a cierto indicativo. Por ejemplo, para obtener un balance deberemos disponer del fichero de cuentas ordenado por el código en orden creciente.

Un fichero puede ordenarse por un indicativo o por varios. Por ejemplo, un fichero de clientes puede estar ordenado por **provincia** y **Saldo**, en ese orden, lo que significa que lo estará por Provincia, y dentro de cada grupo de clientes con la misma provincia, por Saldo.

Criterio de clasificación: Provincia, Saldo, creciente

| Prov. | Saldo | Prov. | Saldo |
|---|---|---|---|
| 1 | 1000 | 5 | 21000 |
| 1 | 5000 | 9 | 7300 |
| 1 | 15000 | 9 | 8500 |
| 1 | 20300 | 18 | 10000 |
| 1 | 25450 | 18 | 10800 |
| 2 | 10000 | 18 | 12500 |
| 2 | 15000 | 20 | 8900 |
| 5 | 1000 | 20 | 16000 |
| 5 | 1500 | 20 | 17500 |
| 5 | 9800 | 20 | 20000 |

Fig. 6.5.

El **criterio de clasificación** de un fichero será el indicativo o indicativos por los cuales está ordenado el fichero, e indicará asimismo si la ordenación, para cada indicativo es creciente o decreciente. En este aspecto, en indicativos alfanuméricos, el orden relativo de números, letras y símbolos vendrá dado por el código interno, que habrá asignado mayor valor a los números que a las letras o viceversa, así como a los posibles símbolos utilizados.

En cierto tipo de ficheros se puede obtener el orden necesario a pesar de hallarse físicamente desordenado. Esto sucede en los ficheros ubicados en soportes direccionables con organización aleatoria, en la que la dirección física de un registro dado se determina en función del valor de un indicativo. De tal forma, asignando sucesivos valores al indicativo, y aplicándoles dicha función, se pueden ir obteniendo los registros en orden, pese a su distribución física al azar. (Ver 6.6.2)

## 6.5. Operaciones sobre ficheros y registros

La vida de un fichero comienza cuando con cierto número de registros se define este fichero; es su **creación**, y la primera operación que sufrirá un fichero. A lo largo de su vida sufrirá otras operaciones:

- **Consulta.** Se accede a uno o varios registros para conocer su contenido.
- **Actualización** o puesta al día. Incluir registros nuevos, suprimir algunos y modificar otros.
- **Clasificación.** Alterar el orden de los registros de acuerdo a un criterio determinado.
- **Reorganización.** Copiar el fichero, para conseguir volver a una estructura inicial que ha ido cambiando por sucesivos tratamientos.

En cuanto a las operaciones sobre registros, pueden ser:

- **Consulta** del contenido de un registro.
- **Inserción** de un registro nuevo en el fichero.
- **Supresión** de un registro existente.
- **Modificación** de su contenido.

## 6.6. Organización de ficheros

### 6.6.1. Introducción

Existen cinco organizaciones de ficheros básicas, de cuya combinación se derivan multitud de organizaciones posibles.

Estas organizaciones básicas son:

- Aleatoria
- Secuencial
- Secuencial indexada
- Secuencial encadenada
- Secuencial indexada–encadenada

En cada caso, se escogerá una organización u otra, dependiendo de las características de los soportes y del **modo de acceso** requerido.

Se habla de **acceso secuencial** cuando se van accediendo posiciones físicas (registros físicos) sucesivas, es decir, cuando tras acceder a la posición N se accede a la N + 1. Se habla de **acceso directo** (o aleatorio) cuando se accede directamente a la posición deseada, sin necesidad de acceder a las posiciones que le preceden físicamente.

### 6.6.2. Ficheros secuenciales y aleatorios

La organización de un fichero viene influida por su modo de creación. **Fichero secuencial** será aquél creado secuencialmente, un registro físicamente detrás del anterior. Normalmente los registros han sido ordenados por algún indicativo, de tal forma que el fichero resultante conserva dicho orden.

Un fichero de este tipo podrá estar soportado en un dispositivo secuencial (cinta magnética, cassette), o en uno direccionable (disco, tambor, diskette).

**Fichero aleatorio** será aquél en cuya creación se ha utilizado acceso aleatorio o directo. Así, al crear el fichero se determina la posición de cada registro por medio de una función, o serie de operaciones sobre el valor de cierto indicativo. En creación los registros podrán estar ordenados o desordenados, lo normal es que sobre el soporte queden desordenados, dependiendo ello de la función de acceso utilizada.

Un fichero aleatorio solo podrá estar ubicado en un soporte direccionable.

Mientras un fichero secuencial simple, normalmente, sólo puede ser utilizado secuencialmente, uno aleatorio permite ser utilizado secuencial o aleatoriamente, aunque por el hecho de estar generalmente en desorden su utilización secuencial no sea usual.

La principal ventaja de los ficheros secuenciales es sin duda su mínima ocupación, ya que ocupan únicamente el espacio en soporte preciso para su tamaño, mientras

que un fichero aleatorio, en razón de la función de asignación, deja huecos vacíos entre registros, ocupando por ello más espacio del necesario en secuencial.

La consulta de un registro en un fichero secuencial puede llegar a ser muy lenta, ya que dependerá de si el registro buscado se halla al principio o al final del soporte; en acceso directo se efectuará en un tiempo que oscila de milisegundos a décimas de segundo. Ello no implica que siempre sea mejor éste acceso, ya que si debemos consultar todos los registros del fichero, o un gran número de ellos, puede resultar más rentable el acceso secuencial, ya que el tiempo de acceso al siguiente en secuencia siempre es mucho menor que el tiempo medio de acceso directo.

En el caso de tener que actualizar (inserción, modificación y supresión) un fichero con frecuencia, habrá que tener en cuenta que mientras en un fichero aleatorio estas operaciones son muy rápidas, en uno secuencial es obligada la duplicación por copia; se va creando un fichero nuevo a partir del existente, en el cual suprimiremos los registros de baja e insertaremos los de alta. Por supuesto, el fichero de movimientos debe seguir el mismo orden del fichero secuencial.

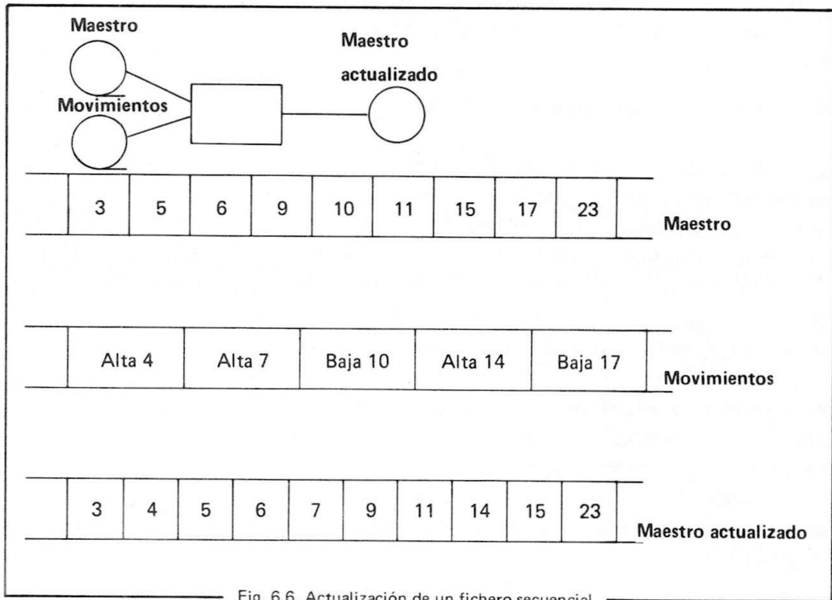


Fig. 6.6. Actualización de un fichero secuencial.

Uno de los mayores problemas que se plantean en un fichero aleatorio es la aparición de **sinónimos** o registros a los que corresponde la misma dirección.

Dada una función de asignación, distintos valores del indicativo pueden originar el mismo resultado, la misma dirección. Aún en el caso de que en la misma direc-

ción se pueden almacenar secuencialmente varios registros, pueden llegar registros que encuentren ocupado el lugar en que deberían ser colocados. Ello obligará normalmente a crear zonas de sinónimos o excedentes a las que irán a parar éstos o a colocarlos en direcciones próximas. En búsqueda, al no encontrar el registro en el lugar direccionado, habrá que rastrear secuencialmente dicha zona hasta localizarlo. El sistema puede llegar a ser sumamente complejo, ya que también habrá que prever la longitud de dicha zona, establecer una o varias, prever su agotamiento, etc.

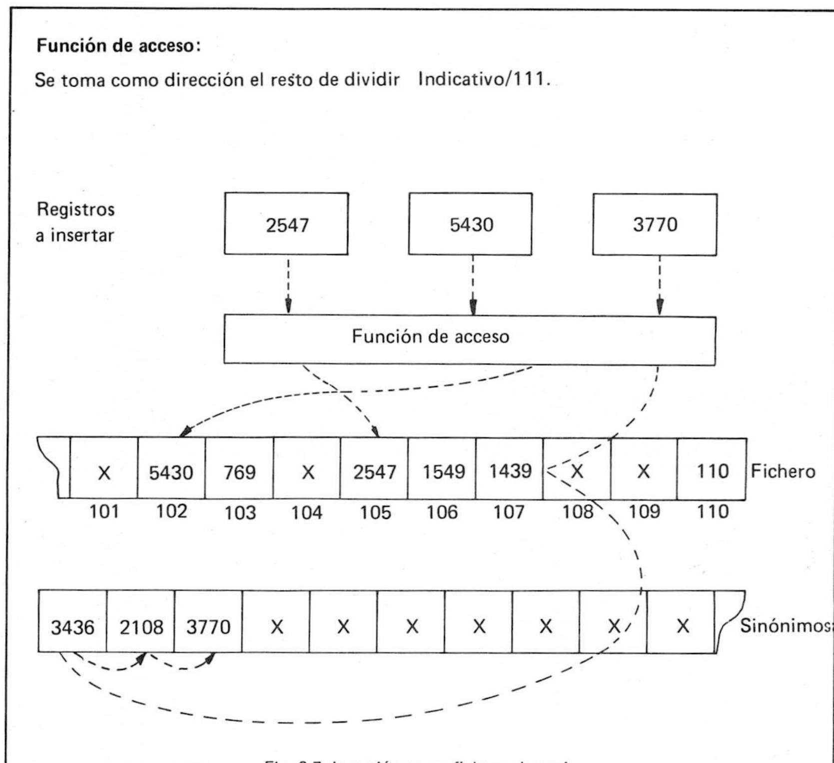



Fig. 6.7. Inserción en un fichero aleatorio.

Las organizaciones secuencial y aleatoria plantean problemas, la primera de rapidez de consulta y la segunda en ocupación de soporte.

Para aliviar estos problemas surgen organizaciones que pretenden mediante diversos métodos disminuir dichos inconvenientes, siempre sobre soportes direccionales. Los métodos utilizados son índices (indexación) y enlaces (encadenamientos). La utilización de índices en una organización secuencial da lugar a una organización **«secuencial-indexada»**, la utilización de encadenamientos a una **«secuencial-encadenada»**; por último, la utilización de índices y encadenamientos, a una **«secuencial indexada-encadenada»**.

### 6.6.3. Ficheros secuenciales indexados

Un índice es una tabla que señala el comienzo de un grupo de registros del fichero. El fichero deberá estar ordenado. En consulta, un método consiste en localizar en el índice el indicativo inmediatamente superior o igual al buscado, y este elemento del índice señala el comienzo de un grupo de registros entre los cuales estará el buscado. La búsqueda en dicho grupo se efectuará en secuencia.

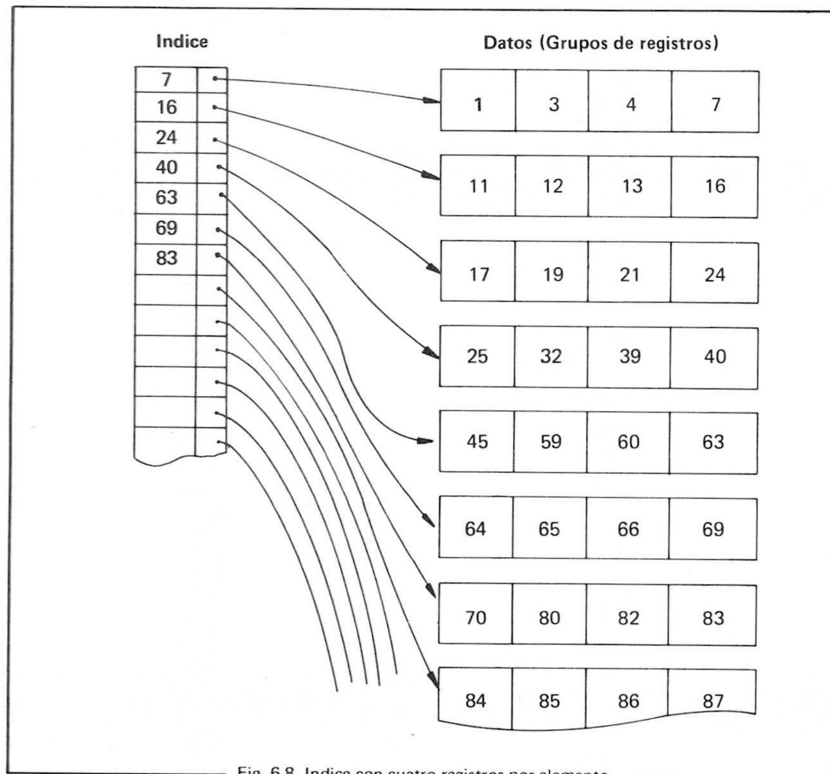



Fig. 6.8. Indice con cuatro registros por elemento.

Del número de registros por elemento dependerá la velocidad de búsqueda. Lo más rápido será un elemento para cada registro (Indice total); en este caso el fichero no necesita estar ordenado, pero la ocupación del índice será muy grande. Generalmente el grupo de registros se asocia al tamaño de la unidad física de lectura del dispositivo, (registro físico), es decir, una o varias pistas, sectores, etc., de tal forma que la búsqueda secuencial dentro del grupo pueda efectuarse en memoria fácilmente.

En ficheros muy grandes puede ser que la búsqueda en el índice se efectúe por medio de otro índice (índices jerarquizados o en árbol).

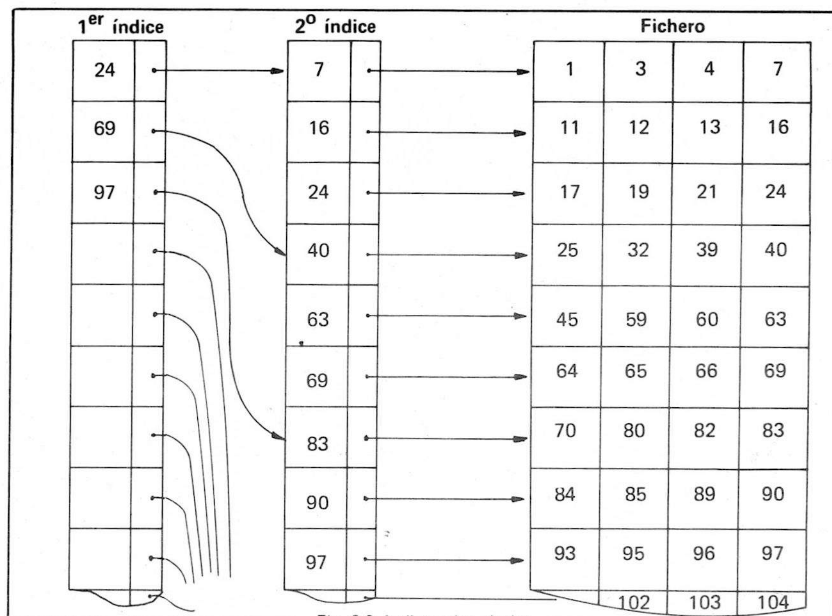


Fig. 6.9. Indice a dos niveles

Dependiendo de la evolución del fichero puede ser necesario reorganizarlo periódicamente; las inclusiones y supresiones de registros precisarán eventualmente la alteración del índice.

Los ficheros indexados no son de acceso tan rápido como los aleatorios, al tener que efectuar búsquedas en el índice y en el grupo de registros, pero en cada caso concreto permiten una optimización variando el número de índices o de registros por elemento, además de no desaprovechar tanto espacio.

### 6.6.4. Ficheros secuenciales encadenados

Para evitar en un fichero secuencial la necesidad de duplicación para insertar o suprimir registros puede utilizarse la organización secuencial encadenada.

Para ello se incluye en cada registro un campo de enlace. En dicho campo pondremos la dirección del siguiente registro (físicamente, puede ser colocado muy lejos del anterior). De esta forma, para incluir un nuevo registro se usará cualquier lugar libre aunque no esté cercano al lugar que le correspondería.

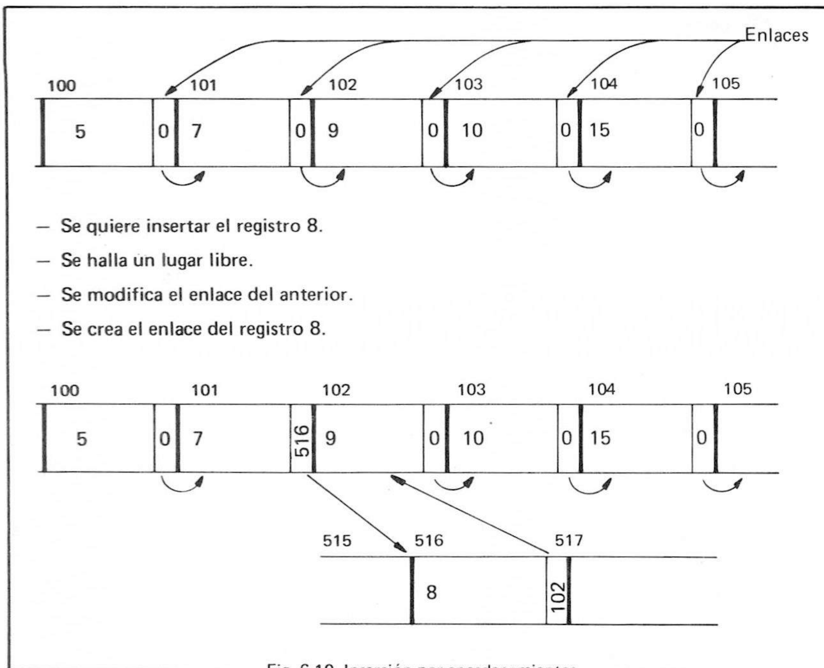


Fig. 6.10. Inserción por encadenamientos

Para consultar un fichero de este tipo se irán siguiendo los enlaces. Si un enlace tiene un valor cero, el siguiente lógicamente será también el físicamente posterior. En otro caso, se irá a la dirección indicada por el enlace.

La supresión de un registro se efectúa variando enlaces, de igual forma que la inserción.

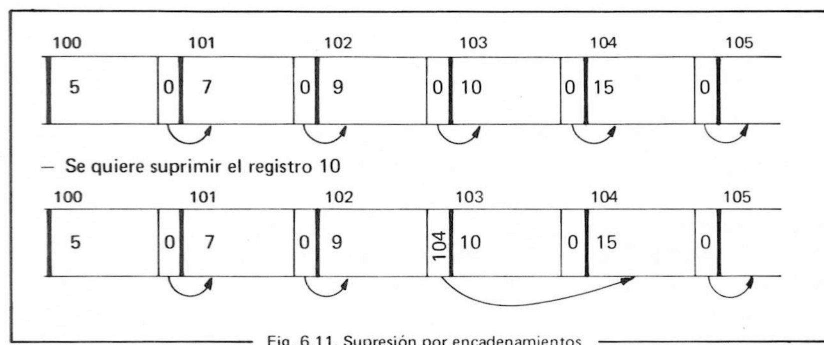


Fig. 6.11. Supresión por encadenamientos

Este tipo de ficheros es de muy cómodo manejo, con el inconveniente de ocupar el fichero más espacio debido a tener que incluir un enlace en cada registro. Por otra parte, con el tiempo, tras muchas inserciones y supresiones, el fichero puede llegar a tener muchos enlaces, resultando muy lento su proceso, por lo que cada cierto tiempo convendrá efectuar un duplicado eliminando dichos enlaces, volviendo a dejar el fichero totalmente en secuencia. Es lo que se conoce como regeneración o reorganización del fichero, al igual que sucedía con los ficheros indexados.

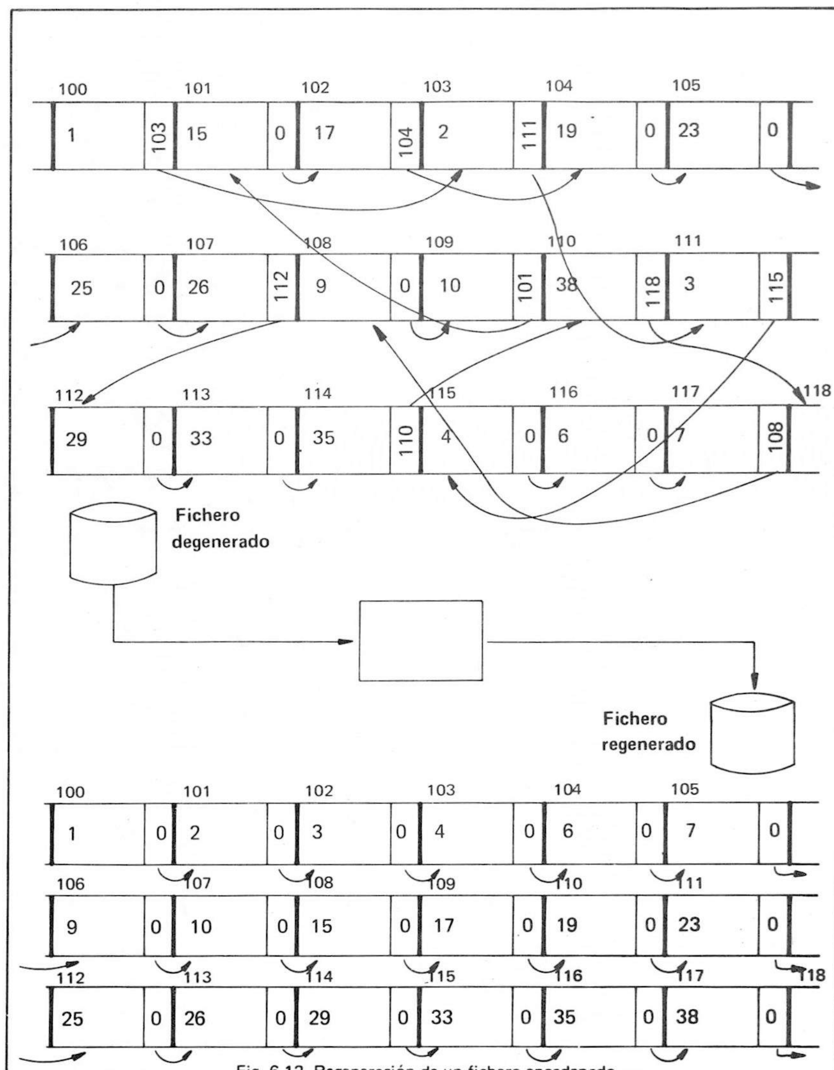


Fig. 6.12. Regeneración de un fichero encadenado

### 6.6.5. Ficheros secuenciales indexado-encadenados

Dotando a un fichero indexado de la posibilidad de utilizar encadenamientos se facilitan las inserciones y supresiones de registros.

Este tipo de fichero será de muy cómoda utilización, pero degenerará más rápidamente que los anteriores, además de ocupar mucho espacio, ya que precisará almacenar índices y enlaces.

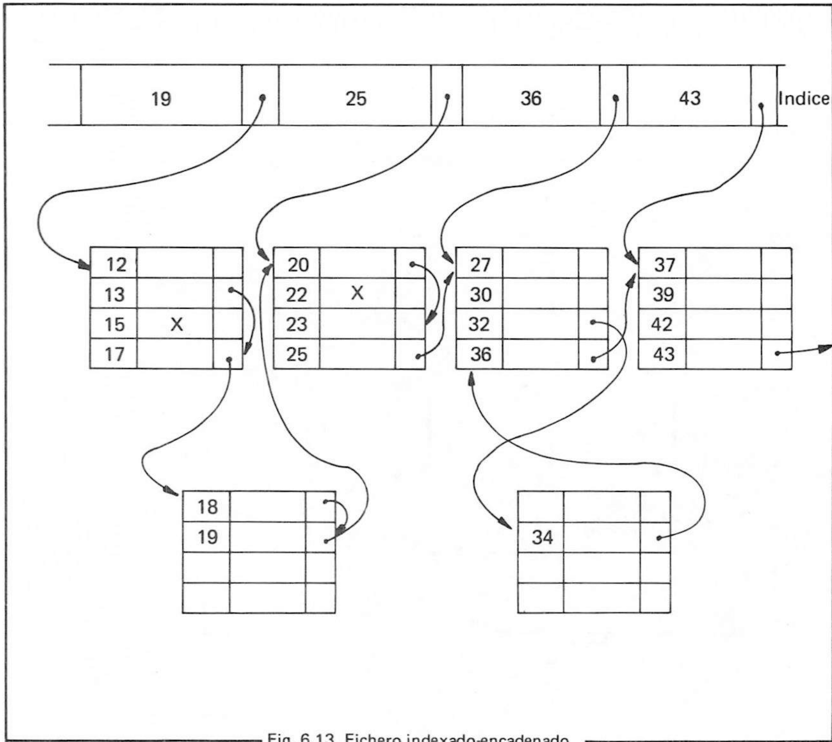



Fig. 6.13. Fichero indexado-encadenado

## 6 7. Utilización de ficheros

### 6.7.1. Acceso secuencial

Es el acceso al fichero por posiciones físicas sucesivas. En creación se grabará y en consulta se accederá al primer registro en la primera posición del soporte que pertenezca al fichero de que se trate, y los demás sucesivamente. En un soporte secuencial, como una cinta magnética, es el dispositivo el que nos obliga a dicho acceso secuencial, mientras que en uno direccionable, por ejemplo un disco, se accederá físicamente siguiendo el orden lógico que proporciona el hardware del soporte:

—Todos los sectores de una pista.
—Todas las pistas de un cilindro.
—Todos los cilindros del disck-pack.

### 6.7.2. Acceso directo

El acceso directo a un fichero es aquél que permite acceder a determinado registro sin pasar por otros, directamente. Para ello, en creación y consulta se utilizará una función de acceso determinada (ver 6.6.1.) que obtiene la dirección física del registro en el fichero a partir del valor del indicativo. La función de acceso más elemental sería hacer que la dirección sea el resultado de multiplicar el valor del indicativo por el tamaño del registro. Así, si cada registro tiene 6 bytes y el valor del indicativo es 120, su dirección sería 6 x 120 = 720 (6 bytes a partir del 720), pero esto plantea el problema de los **huecos**. Un fichero de 7.000 clientes puede llegar a necesitar 50.999 registros si asignamos 2 dígitos para provincia y otros 3 de numeración secuencial (habría que prever lugar para el cliente 999 de la provincia 50, y sin embargo, en el fichero quedarían 50.999–7.000 = 43.999 huecos, lugares vacíos).

Por esta razón habrá que buscar funciones de acceso que eliminen o disminuyan dicho problema. El método ya visto será utilizable únicamente cuando por la naturaleza del indicativo se den muy pocos huecos. Otro método puede ser mantener una tabla con los lugares ocupados y libres, y asignar al indicativo el primer lugar libre. Esto sería mantener un índice total (ver 6.6.2.) pero tiene el inconveniente de precisar sitio para dicho índice, aparte del tiempo de búsqueda.

Si la codificación deja huecos conocidos y de cierto tamaño, es posible acortar el indicativo. Si sabemos que no hay códigos desde el valor 1.500 al 2.000, asignaremos al código 2.001 el valor 1.501 y así sucesivamente.

**Ejemplo.–**
Fichero de 13.000 registros
Huecos: 1.501 a 2.000, 5.001 a 8.500, 12.001 a 15.000

| Código original | Código asignado | Operación |
|---|---|---|
| 1 a 1.500 | 1 a 1.500 | – |
| 2.001 a 5.000 | 1.501 a 4.500 | Restar 500 |
| 8.501 a 12.000 | 4.501 a 8.000 | Restar 4.000 |
| 15.001 a 20.000 | 8.001 a 13.000 | Restar 7.000 |

Otro método de eliminar huecos es suprimir cifras del código. Esto puede hacerse despreciándolas sencillamente o bien obteniendo un número de menos cifras por operaciones sobre las cifras del primitivo.

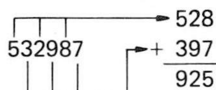
**Ejemplo.–**
Supresión de cifras $2^a$, $3^a$ y $5^a$

| Código original | Código asignado |
|---|---|
| 53298742 | 59742 |
| 39482101 | 38101 |
| 41593400 | 49400 |
| 59872236 | 57236 |

**Ejemplo.–**

Sumar cifras $1^a$, $3^a$ y $5^a$ con $2^a$, $4^a$ y $6^a$

| Código original | Código asignado |
|---|---|
| 532987 | 925 |
| 394821 | 1323 |
| 415934 | 647 |
| 598722 | 1554 |




Otro método, el más usual, consiste en dividir el indicativo por el número primo más próximo e igual o mayor al número de direcciones del soporte que se desea reservar para el fichero (o simplemente por el número de direcciones) y tomar el resto. Así en un fichero con 5.000 registros se dividiría por 4.999. Los restos serán números entre 0 y 4.998, y además saldrán uniformemente repartidos, es decir, no es previsible estadísticamente que no salga nunca el resto 815, así como tampoco que éste se dé con más frecuencia que otros.

Todos los métodos anteriores coinciden en un inconveniente, la aparición de sinónimos (ver 6.6.1.). Aprovechando la noción de bloque o registro físico vista en 6.1

suele dividirse el fichero en **grupos** del tamaño de uno o varios bloques; entonces, la dirección que se obtiene por la función de acceso es la del grupo, y en él se hallarán todos los registros sinónimos. Dentro del grupo se buscará el registro en secuencia. Si aparecen más sinónimos de los que caben en un grupo habrá que crear un enlace a otro grupo que tenga sitio libre. Si, necesitando 1.000 grupos, se asignan al fichero 1.200, el número de sinónimos disminuirá, y con ello el tiempo de búsqueda, a costa de precisar más espacio en el soporte; en cualquier caso se puede llegar a un punto en que ventajas e inconvenientes se equilibren mutuamente, lo cual suele ocurrir para un **factor de carga** (espacio necesario/espacio asignado) del orden de 0,7.

En cualquier sistema de direccionamiento de los estudiados aparecen tres tipos de dirección.

- **Dirección absoluta.** El número del primer registro físico en que se halla el registro lógico buscado.
- **Dirección relativa.** El número del registro físico en que se halla el registro lógico buscado, contando desde el principio del fichero, no del soporte.
- **Dirección lógica.** El número del registro lógico desde el principio del fichero. También, si se utilizan grupos, el número de grupo desde el comienzo del fichero.

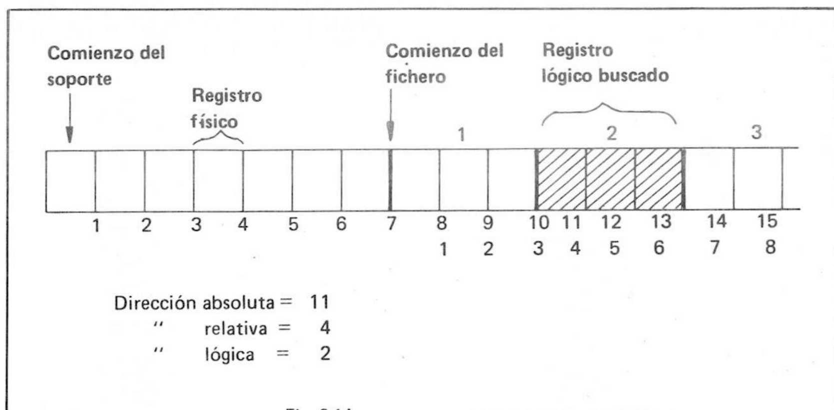


Fig. 6.14.

Generalmente se trabaja con direcciones lógicas, ya que los ordenadores suelen disponer de sistemas que se encargan de convertir éstas en absolutas, que son aquéllas que el equipo precisa, conociendo el inicio del fichero y el tamaño del registro lógico.

### 6.7.3. Ficheros secuenciales

a) **Consulta.** Sobre soportes no direccionables sólo podrá hacerse rastreando secuencialmente el fichero hasta hallar el registro buscado. En soportes direccionables, si la codificación no tiene huecos, podrá calcularse la dirección del registro buscado:

$$\text{Dir} = \text{Inicio fichero} + (\text{Valor indicativo} - 1) \times \text{Long. registro}$$

Si la codificación tiene huecos pero el fichero está ordenado puede aplicarse la búsqueda **por dicotomía**. En este sistema se accede a la mitad del fichero y se compara el indicativo del registro hallado con el buscado; si el hallado es mayor, el buscado estará en la mitad izquierda, si es menor, en la derecha; entonces se accede a la mitad de la parte correspondiente, efectuando el mismo proceso, hasta hallar el registro buscado. El número máximo de consultas según puede deducirse matemáticamente de forma sencilla, será $\log_2 N$, siendo N el número de registros del fichero.

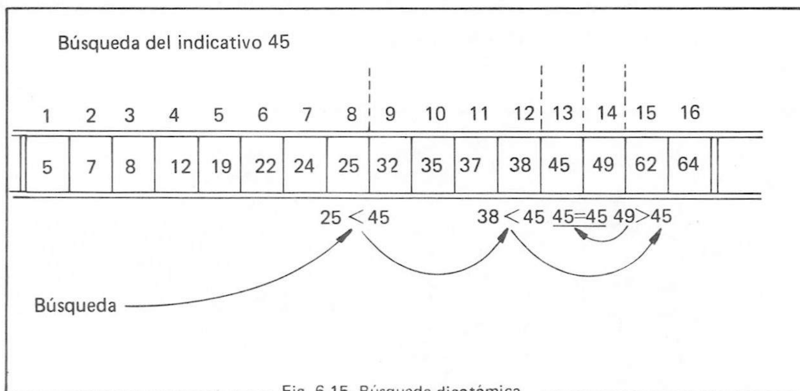


Fig. 6.15. Búsqueda dicotómica.

En caso de ficheros desordenados sólo será posible la búsqueda secuencial.

b) **Actualización.** En soportes no direccionables cualquier actualización requerirá la copia del fichero, ya que incluso en el caso de que el dispositivo permitiese modificar bloques dentro de una cinta, por ejemplo, los programas del Sistema no suelen prever tal posibilidad. La inserción de registros en medio de un fichero es imposible en cualquier tipo de soporte, debiendo hacerse por copia o por inclusión al final del fichero.

En ficheros situados sobre soportes direccionables es posible la modificación de registros y la inserción al final del fichero, pudiendo suprimir registros por marca, colocando en algún lugar del registro un código que indique que el registro está anulado.

c) **Clasificación.** Una clasificación es un tipo especial de actualización. Por ello, según lo visto en el apartado b), clasificar una cinta requerirá su traspaso a otro u otros soportes, sobre los cuales se efectuará dicha ordenación. La clasificación podrá efectuarse sobre el propio fichero o bien creando uno nuevo, según la técnica utilizada.

d) **Reorganización.** Será precisa en el caso de anulación de registros por marca, ya que dichos registros seguirán ocupando su lugar en el soporte.

### 6.7.4. Ficheros aleatorios

a) **Consulta.** Se efectuará aplicando el método de direccionamiento utilizado al crear el fichero. Una vez calculada la dirección del grupo se cargará éste en memoria, buscando el registro en secuencia; si no estuviera en dicho grupo habrá que seguir los enlaces creados para los sinónimos, hasta encontrarlo. En ciertos casos puede ser conveniente tratar el fichero en secuencia.

b) **Actualización.** Modificar un registro supone cargar en memoria el grupo correspondiente, efectuar la modificación y volver a grabar el grupo en el soporte. La inserción de un registro requerirá buscar sitio en su grupo o al final de la cadena de excesos, colocar el nuevo registro y volver a grabar. En cuanto a la supresión, se podrá hacer por marca o realmente; por marca será idéntico proceso al de la modificación, mientras que al suprimirlo realmente habrá que reorganizar el grupo, dejando sitio libre al final, o bien modificar los enlaces si se halla en zona de excesos.

c) **Clasificación.** Clasificar un fichero aleatorio supone obtener otro organizado secuencialmente con la misma información, lo cual se llevará a cabo por copiado, accediendo al fichero aleatorio por valores crecientes de algún indicativo o indicativos.

d) **Reorganización.** Se llevará a cabo para eliminar el espacio ocupado por registros anulados por marca. Si el fichero degenera por ir aumentando los sinónimos, la reorganización implicará el cambio de la función de acceso o el aumento de tamaño de los grupos.

### 6.7.5. Ficheros secuenciales encadenados

a) **Consulta.** Se seguirán los enlaces hasta hallar el registro buscado, pudiendo también tratarse el fichero secuencialmente.

b) **Actualización.** Modificar un registro implicará localizarlo, cargarlo en memoria y volverlo a grabar en su lugar, ya modificado. La inserción y supresión de registros se efectuará modificando los enlaces (ver (6.6.3.).

c) **Clasificación.** Un fichero encadenado siempre debe estar ordenado por algún indicativo. Para ordenarlo por cualquier otro habrá que crear un nuevo fichero sobre el mismo lugar o creándolo en otra posición.

d) **Reorganización.** Será necesaria para disminuir el tiempo de consulta cuando hay muchos enlaces (ver 6.6.3.).

### 6.7.6. Ficheros secuenciales indexados e indexado-encadenados

a) **Consulta.** Se buscará en el índice, en secuencia o dicotómicamente, el grupo en que se halla el registro buscado, y luego se busca en secuencia dentro del grupo, siguiendo los encadenamientos si los hay. (Fig. 6.13.). También puede efectuarse un tratamiento secuencial.

b) **Actualización.** La modificación de registros no plantea problemas. En cuanto a inserción y supresión harán necesaria la modificación de los enlaces si los hay y/o del índice. La supresión puede efectuarse también por marca.

c) **Clasificación.** Si se quiere clasificar el fichero por algún indicativo distinto del actual será necesario crear otro fichero con su índice.

d) **Reorganización.** Periódicamente se eliminarán los enlaces y registros anulados por marca. A partir de las estadísticas de utilización se podrá llegar a una reorganización que altere el número de índices o de registros por elemento.

## 6.8. Bases de Datos

### 6.8.1. Concepto

Las Bases de Datos o Bancos de Datos son **grandes conjuntos de información interrelacionada.** Su utilización es cada día mayor, y pueden considerarse como el último paso en la creación de estructuras de información. El objetivo de una Base de Datos es en último término el disponer de una estructura de manejo cómodo, que evite la gestión de múltiples ficheros individuales, muchas veces con información duplicada y cuyo mantenimiento es muy laborioso. Una Base de Datos es en definitiva un conjunto de ficheros, llamado a veces **fichero integrado,** pero la Base de Datos posee un **Sistema de Gestión** que evita al usuario el conocimiento de su estructura interna, apareciendo para él como una estructura más sencilla de la que realmente posee. La principal característica de una Base de Datos es la interrelación existente entre los datos que la componen, que permite acceder a un mismo dato por múltiples caminos, por varios indicativos distintos.

Podemos imaginar a título de ejemplo una Base de Datos en la que se mantenga el censo de habitantes de una ciudad.

En ciertos casos interesará averiguar los datos de un ciudadano accediendo a él por su número de Documento Nacional de Identidad. Para ello, la Base de Datos deberá disponer de un índice por este indicativo.

De la misma forma puede ser necesario acceder a la información por el Nombre o por el Domicilio, lo cual obligaría a disponer de otros índices por estos indicativos, y esto puede ser necesario para otros muchos.

Al mismo tiempo serán muy usuales las consultas a efectos estadísticos, por ejemplo preguntando por el número de habitantes varones, mayores de 25 años, que tienen a su cargo más de 2 personas. Todos estos datos deberán estar almacenados en la Base de Datos, y para evitar búsquedas secuenciales de la totalidad, puede ser necesario encadenar entre sí, por ejemplo, todos los ciudadanos varones, o todos los mayores de cierta edad, dependiendo de la frecuencia de consulta por dicho indicativo, de forma que dada la condición **mayores de 25 años**, la búsqueda se efectuará siguiendo estos enlaces, accediendo únicamente a los registros que cumplen dicha condición.

Físicamente, tal como se muestra en la figura 16, puede crearse un fichero indexado por el nombre, un índice por número de registro, crear enlaces entre los individuos del mismo sexo y otros enlaces, sobre el campo Edad, uniendo los registros por edades entre 0 y 25 años, entre 25 y 50 y de más de 50 años.

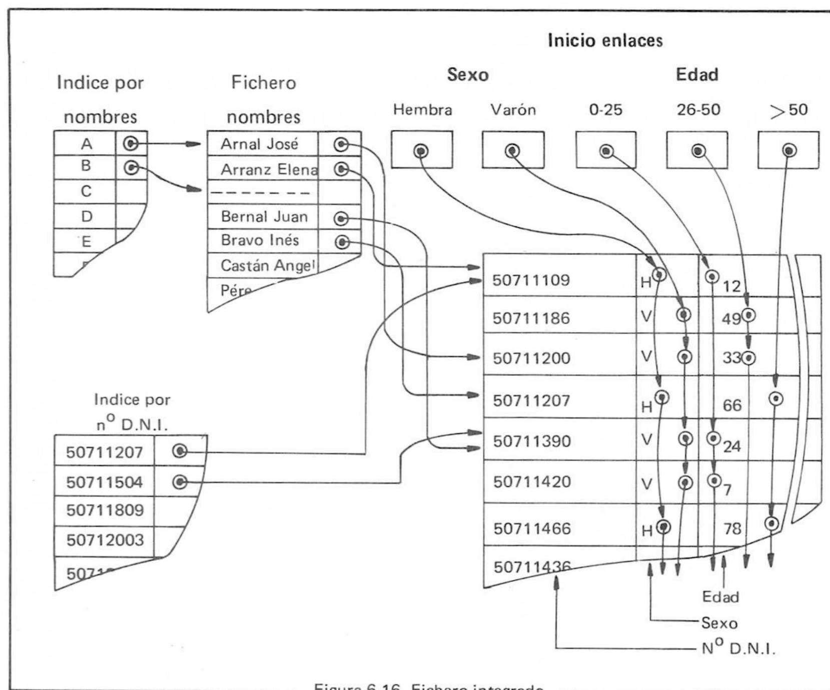



Figura 6.16. Fichero integrado

Se observa que los métodos utilizados para la creación de un Sistema de Gestión de Bases de Datos son los mismos utilizados en ficheros, esto es, **índices y encadenamientos**; el método varía mucho de unos sistemas a otros, existiendo Bases de Datos en las que desaparece totalmente el concepto clásico de fichero para reducirse a un gran conjunto de datos interrelacionados.

En algunos sistemas se separan totalmente los datos de los enlaces; existen por un lado los ficheros de datos, y por otro un **modelo** (organizado en forma de árbol u otra organización compleja) en el que se representan los indicativos y enlaces hacia los ficheros de datos, de forma que las búsquedas y seguimiento de enlaces se efectúan en el modelo estructural, accediendo a los datos (en memorias lentas de gran capacidad) una vez localizada la dirección buscada.

En la figura 17 se representa un esquema de una Base de Datos en la cual un fichero de almacén, otro de clientes y otro de pedidos se interrelacionan. Este esquema podría responder fácilmente a consultas como:

– Listar todos los pedidos del cliente x
– Listar los productos del pedido x
– Listar los clientes que han pedido el producto x
– etc.

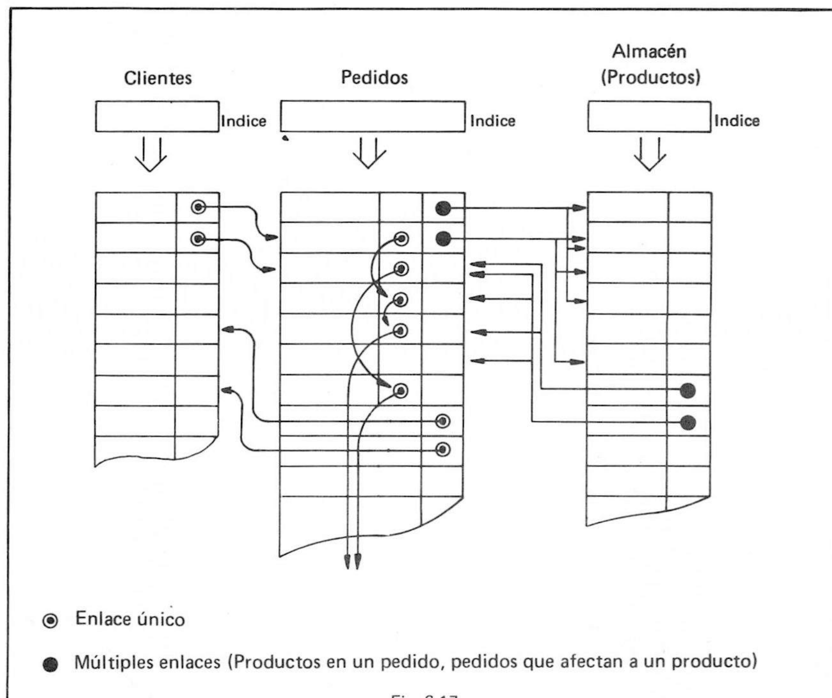


Fig. 6.17.

### 6.8.2. Estructura de una Base de Datos

Se distingue entre **estructura lógica** y **estructura física**. La primera es la que el usuario define y por la que se rige para efectuar consultas, y la segunda es la que realmente posee la Base de Datos, y será mucho más compleja que la primera, ya que incluirá índices, encadenamientos, etc, que el usuario no tiene por qué conocer.

Existen tres estructuras lógicas básicas: **lineal**, en **árbol** y en **red**. Un fichero secuencial poseería una estructura lógica lineal, como puede ser un fichero de movimientos; el fichero de componentes de un fabricado tendrá estructura en árbol, y en el ejemplo visto en la figura 17 aparece una estructura en red. Lo normal es que en una Base de Datos se encuentren mezclados los tres tipos formando una estructura compleja.

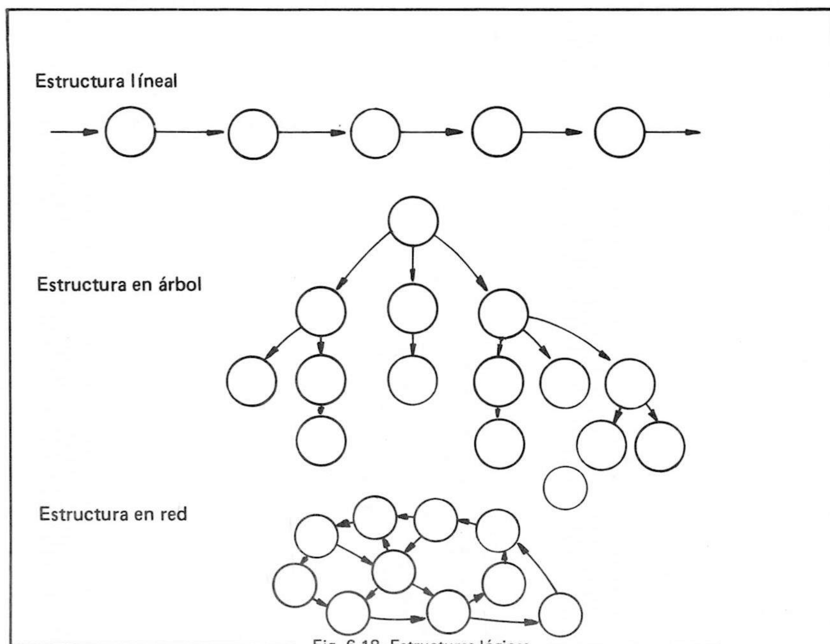



Fig. 6.18. Estructuras lógicas.

En la creación se procurará establecer dicha estructura de forma que el acceso más frecuente sea el más rápido; por ejemplo, si se quiere acceder por el N° de registro con mucha frecuencia y por Nombre tan sólo eventualmente, quizá se construya un índice total por N° de registro, efectuando la búsqueda por Nombre a base de un índice no total, o por encadenamientos, etc. En este caso, N° de registro sería el indicativo **primario**, y Nombre uno de los posibles indicativos **secundarios.**

Lo normal es que se permita acceder al fichero y al registro, y no a elementos menores de la Base de Datos.

### 6.8.3. Relaciones entre datos

En una Base de Datos se querrán establecer relaciones **entre ficheros** y relaciones **entre los datos** de un mismo fichero.

Dentro del fichero visto en la figura 16, se establecían enlaces entre los elementos del fichero que pertenecían al mismo grupo de edades y entre los que tenían el mismo sexo. En el ejemplo de la figura 17 existían relaciones entre los ficheros de Clientes, Pedidos y Almacén.

En estos casos la relación se efectuaba por medio de enlaces, pero dependiendo del sistema, de la frecuencia de consulta del indicativo, del número de elementos, etc, puede ser más rentable establecerlas por medio de índices, tablas u otros métodos.

Las relaciones entre elementos de un conjunto de datos determinado pueden ser de tres tipos:

- Relación **dueño-miembro** (Desde un elemento se enlaza con sus elementos subordinados).
- Relación **miembro-dueño** (Desde un elemento se enlaza con el elemento superior, al que está subordinado)
- Relación **miembro-miembro** (Enlace de un elemento con otro de su rango, ambos subordinados a un mismo elemento superior)

**Ejemplos.–**

Dueño-miembro. El registro de un cliente señala a todos los registros de pedidos por él efectuados.

Miembro-dueño. Un registro de pedido señala al registro del cliente al que pertenece.

Miembro-miembro. Los pedidos de un mismo cliente se enlazan entre sí.

### 6.8.4. Lenguajes de Bases de Datos

- **Lenguaje de descripción.** Suele existir un lenguaje mediante el cual el usuario puede definir la estructura de la Base de Datos. Lo normal es que se describan los datos por sus nombres, longitudes, etc, y sus interrelaciones, es decir, la estructura lógica, a partir de la cual el sistema configura la estructura física adecuada.

- **Lenguaje de programación.** Será el lenguaje en el cual el usuario programará las aplicaciones que utilicen la Base de Datos, estableciendo comunicación con ella desde el programa. Puede ser un lenguaje específico o una serie de instrucciones que amplían un lenguaje ya existente (COBOL, BASIC), instrucciones que se conocen en ese caso como huéspedes.

- **Lenguaje de consulta.** Suele ser un intérprete de manejo muy sencillo para efectuar consultas a la Base de Datos, utilizable por personal sin formación informática, para ser usado desde terminales.

- **Editor de informes.** Usado para obtener listados, estadísticas, etc, incluirá funciones de descripción de cabeceras, totales, cálculos sencillos, selección de sólo ciertos registros por medio de condiciones lógicas, etc.

# CAPITULO 7. SISTEMAS OPERATIVOS

## 7.1. Concepto de Sistema Operativo

La evolución en los equipos electrónicos ha sido grande y la complejidad ha ido aumentando. Para conseguir una mayor facilidad en la utilización y un mayor rendimiento de la máquina, los fábricantes de ordenadores, actualmente, suministran:

1) Conjuntos de programas, llamados **programas de control**, para facilitar, automatizar y mejorar el rendimiento en los procesos de explotación. Programas que encadenan automáticamente los distintos trabajos o programas, otros que consiguen simultaneidad de operación entre distintos órganos de la máquina; otros que tratan errores, etc. También se le suele llamar **sistema de explotación**.

2) Conjuntos de programas, llamados **programas de servicio**, para mejorar la productividad de los programadores facilitando su labor. Los programas traductores (ensambladores y compiladores) y las rutinas de utilidad (programas para clasificar, copiar ficheros, probar programas, etc.).

A estos dos conjuntos de programas, o sea al conjunto formado por los programas de control y los programas de servicio, es a lo que se llama **sistema operativo.**

Aunque hemos dicho que estos programas los suministra el constructor, el usuario puede ir después incorporando programas propios e ir aumentando la capacidad de su sistema operativo, aunque esto requiere en general, y sobre todo en los programas de control, un gran conocimiento del sistema operativo y una técnica muy avanzada.

Interesa destacar que algunos también incluyen en el sistema operativo, concretamente en los programas de servicio, los programas de las aplicaciones del usuario. Este es el concepto más amplio de sistema operativo. En cambio, con frecuencia, otros lo toman en un sentido más restringido y entienden por sistema operativo sólo los programas de control.

De esta última forma el AUERBACH EDP Reports define el sistema operativo como:

"Colección ordenada de rutinas y procedimientos que acompañan al ordenador y normalmente realizan algunas o todas de las funciones siguientes:

- Planificación, carga, iniciación y supervisión de la ejecución de programas.

- Asignación de memoria, unidades de E/S y otros dispositivos del sistema.

- Iniciar y controlar todas las operaciones de E/S.

- Manejar errores y reiniciaciones.

- Coordinar las comunicaciones entre el operador y el sistema.

- Mantener un diario de las operaciones del sistema.

- Controlar las operaciones cuando se trabaja en los modos de multiprogramación, multiproceso y time-sharing."

Hechas estas salvedades, vamos a considerar el sistema operativo en su concepción más amplia, para así tener una visión más general. En esta concepción el Software o conjunto de todos los programas está formado por el sistema operativo más los programas independientes que no se incluyen en ninguno de los tipos citados.

## 7.2. Opciones de los Sistemas Operativos

### 7.2.1. Memoria Virtual

En ordenadores sin memoria virtual, el tamaño de los programas viene limitado por el de la Memoria Central. Esto puede suponer un grave problema, y aún más cuando, como veremos al hablar de Multiprogramación, se necesita tener varios programas en memoria.

Para evitar este contratiempo sin necesidad de aumentar la capacidad del ordenador, que lo encarecería sensiblemente, se procede al uso de las **segmentaciones**, que son realizadas por los compiladores, es decir, se dividen los programas en trozos **(segmentos, páginas)** de un cierto tamaño, y se estructura la Memoria Central en trozos **(áreas de solapamiento)** sobre las que se irán cargando los trozos de programa que se vayan necesitando.

De esta forma, aunque un programa ocupe más lugar del espacio real de memoria, el S.O. lo va pasando a memoria a trozos, en cada momento el segmento de programa que está activo, ejecutándose así sin problemas.

Por supuesto, esto implica que la traducción de los compiladores se haga a base de **direccionamientos relativos,** ya que un segmento puede ser cargado en cualquier área de memoria. Al ir ejecutando las instrucciones, y conociendo en dónde se ha cargado el segmento, el equipo añadirá a la dirección relativa el comienzo del área de trabajo, convirtiéndola en absoluta.

Evidentemente, con este sistema podemos tener en memoria varios programas al tiempo (varios segmentos de programa).

La segmentación permite, pues, que varios programas en conjunto o separados, que superen la capacidad de memoria de los ordenadores, puedan ser ejecutados de forma tal que el programa monitor carga en memoria en cada instante solamente aquellas rutinas que son activas.

### 7.2.2. Multiprogramación

La multiprogramación surgió con el fin de obtener un aprovechamiento o rendimiento máximo de las distintas unidades que componen un ordenador. Concretamente, la velocidad de la unidad central de proceso con componentes exclusivamente electrónicos es miles de veces superior a la de los periféricos con componentes electromecánicos. De esta forma, en la mayoría de los procesos, la unidad central ha de estar ociosa durante frecuentes intervalos de tiempo en espera de operaciones de entrada/salida.

En realidad, el primer paso para ocupar este tiempo disponible o tiempo de espera del procesador para tener datos, es el empleo de los **canales**, con lo que se

consigue simultaneidad de operación de proceso y de entrada/salida. De esta forma, mientras se está realizando una operación de entrada/salida bajo control de un canal, el procesador puede continuar trabajando y para cuando necesite nuevos datos tenerlos ya en memoria. No obstante, con la diferencia de velocidades indicadas, salvo algunos casos en que las operaciones a realizar con los datos sean relativamente muy numerosas, la unidad central sigue teniendo que esperar por operaciones de entrada/salida. En estos casos el procesador, cuando inicia una operación de entrada/salida, necesita que ésta finalice para poder continuar el proceso, y entonces sigue estando ocioso.

Con la multiprogramación, como hemos dicho, tratamos de conseguir una ocupación máxima de la UCP y se hace posible con la ejecución de dos o más programas de forma concurrente y pasando el control de un programa a otro cuando por el primero tendría que estar en espera.

De hecho, al haber varios programas en memoria, cuando se produce una interrupción debido a la petición de entrada/salida de uno de ellos, se cede el control a otro programa, con lo que la unidad central sigue trabajando con este nuevo programa, mientras que simultáneamente se está efectuando la operación de entrada/salida bajo el control del canal. Si el segundo programa se interrumpe pasa a un tercero, y así sucesivamente.

Para conseguir este modo de operación, que de estar bajo el control del programador sería laboriosísimo, se han construido sistemas operativos evolucionados que controlan todas las operaciones.

Hay diferentes grados de multiprogramación: desde una multiprogramación restringida o grado más elemental, en el que se hacen particiones de memoria fijas a las que se asignan también unidades fijas y funcionando las particiones independientemente con sus correspondientes hileras de programas, hasta el más sofisticado en el que, mediante asignaciones dinámicas, toda la memoria está disponible para los diferentes programas y todas las unidades externas se comparten, pasando por distintos niveles intermedios.

### 7.2.3. Tiempo Compartido

Un paso más evolucionado en la utilización multiprogramada de un ordenador es su utilización en tiempo compartido. Consiste en hacer una utilización simultánea del sistema, al menos a escala macroscópica, por varios utilizadores independientes. Es decir, los respectivos problemas son ejecutados por sus propios programas, e incluso sus unidades de E/S son independientes, aunque también varios pueden acceder a la misma unidad, de manera que trabajan como si el ordenador perteneciera exclusivamente a cada usuario.

Un ejemplo típico de tiempo compartido es la utilización del ordenador para enseñanza simultánea a distintos alumnos (CAI).

Con esta noción de tiempo compartido ya puede verse que es necesaria una multiprogramación, de tal forma que el procesador atienda a los diferentes programas (usuarios) durante ciertos intervalos, pero que, dada su velocidad, da la impresión de tener su utilización exclusiva. El paso de un programa (usuario) a otro se hace por las interrupciones de E/S, de las que ya habíamos hablado en multiprogramación, o por reloj: cuando se ha dedicado un tiempo determinado a un usuario se pasa al siguiente, con objeto de que si alguno requiere un tiempo excesivo no se produzca una monopolización.

El número de programas que pueden estar operando simultáneamente (a escala macroscópica o aparente) aumenta considerablemente. De un máximo de 8 ó 9 (en algún caso extremo, 15) programas en multiprogramación se pasa hasta un máximo del orden de 250 programas en tiempo compartido. Lógicamente, en este último caso, no pueden estar todos los programas simultáneamente en memoria. Se hace un almacenamiento dinámico de programas de tal forma que cualquier programa es reemplazado sin finalizar por otro que se vaya a utilizar en ese instante y que se trae de una memoria externa de acceso directo y muy rápido. Se requieren técnicas muy especializadas.

De cualquier forma, dentro de los sistemas de tiempo compartido, se encuentran diversos grados de sofisticación; desde el tipo de pregunta-respuesta en el que el utilizador sólo puede formular y recibir en su terminal mensajes determinados, hasta el que permite formular, compilar y ejecutar los programas desde cada terminal y en distintos lenguajes.

Con todo lo visto, podemos deducir que cualquier sistema de tiempo compartido incluye la multiprogramación, pero que no queda muy definido a partir de qué punto una multiprogramación avanzada se puede considerar tiempo compartido. La situación de esta frontera varía de unos autores a otros.

### 7.2.4. Multiproceso

Por razones de seguridad (posibles errores o averías) o por razones de potencia o velocidad puede ser interesante utilizar **varios procesadores** funcionando simultáneamente y compartiendo memorias centrales, externas y periféricas. Cuando esto ocurre decimos que tenemos un sistema de multiproceso.

Los trabajos se comparten entre los procesadores y se pasan de unos a otros en caso necesario. El sistema operativo ha de ser capaz de controlar esto con la utilización que, además, estemos haciendo de multiprogramación o tiempo compartido. Obsérvese que, por supuesto, el multiproceso y el tiempo compartido no son excluyentes entre sí, antes al contrario, lo normal es utilizar el multiproceso en sistemas de tiempo compartido.

### 7.2.5. Tiempo Real

En algunos casos surge la necesidad de que un sistema procese una determinada información y nos devuelva los resultados con unas restricciones de tiempo muy fuertes. Si el sistema es capaz de trabajar de esta forma decimos que lo hace en tiempo real.

El decir que el sistema ha de tener los resultados con unas restricciones de tiempo muy fuertes es un término bastante ambiguo. En realidad, el tiempo máximo varía con cada sistema en particular y puede ir desde tiempos en milisegundos, si se ha de registrar la traza de un radar; en segundos, para reserva de billetes o aplicaciones bancarias, decenas de segundos, en control de almacenes, y hasta minutos en algunos sistemas de control de procesos. Para estos últimos hay quien prefiere decir que el sistema trabaja en "tiempo útil", mientras que el término "tiempo real" lo reservan exclusivamente para los primeros.

En todos estos casos vemos de común que los resultados que obtenemos se utilizan para continuar, o influyen en el mismo proceso de los datos que los han originado.

Lo normal es que el envío de información y recepción de resultados se haga en terminales alejados del ordenador y que haya que hacer una transmisión de datos a través de determinadas líneas. En este caso, en el tiempo real va implícito lo que llamamos **teletratamiento** o teleproceso.

También lo normal es que un sistema de tiempo real esté siendo utilizado por varios usuarios y, por tanto, en tiempo compartido. Al mismo tiempo, si se está utilizando un sistema en tiempo compartido lo lógico es que el tratamiento sea en tiempo real.

De aquí el que, sin ser exactamente lo mismo, y sin que siempre coincida, se emplea en muchas ocasiones los términos tiempo real y tiempo compartido en forma indistinta.

Por supuesto, un sistema en tiempo real puede ser también de multiproceso, etc.

Como resumen de lo que hemos visto, podemos indicar que la diferencia fundamental en cuanto a utilización del ordenador la podríamos hacer en que fuese en monoprogramación o en modo multiprogramado, y después ya se irá aumentando la complejidad hardware y del sistema operativo, según se vaya añadiendo una o varias de las posibilidades de tiempo compartido, tiempo real o multiproceso.

## 7.3. Estructura del Sistema Operativo

### 7.3.1. Estructura general

En cuanto a la estructura de los sistemas operativos, hay gran diferencia entre los distintos constructores. Podemos citar dos núcleos o escuelas fundamentales: una que tiende a permitir utilizar la máquina en el modo de explotación que se desee utilizando varios niveles, y otra que tiene sus sistemas operativos ya basados en la multiprogramación.

De todas formas, aunque la orientación sea distinta de una escuela a otra y la estructura también cambie de un constructor a otro, incluso de la misma escuela, las funciones a realizar por un sistema operativo son siempre las mismas, y de acuerdo a esto vamos a ver su estructura. Tampoco nos va a importar que sea un sistema operativo orientado a monoprogramación o multiprogramación. Esto influirá en complejidad y en inclusión de algunas rutinas, pero nada más.

Consideraremos constituido al Sistema Operativo por los programas de control y los programas de servicio, de la siguiente forma:

1) Programas de control:
   a) Programas de gestión del sistema.
   b) Programas de gestión de trabajos.
   c) Programas de gestión de datos.

2) Programas de servicio:
   a) Traductores.
   b) Rutinas de utilidad.
   c) Programas de aplicación.

Cada uno de estos conceptos los estudiaremos por separado en sucesivos apartados. De toda esta serie de programas habrá parte que esté siempre en memoria interna, y se llamará el **residente**, o también **núcleo.** Puede ser puramente software y cargarse como un programa más en memoria, o firmware, y estar permanentemente en memoria especializada reservada para sólo consulta (memorias ROM). Otra parte estará en la biblioteca de programas y se llamará a memoria conforme se vaya necesitando, llevándola también a áreas determinadas.

### 7.3.2. Programas de Control

#### 7.3.2.1. Gestión del Sistema (Supervisor)

A este conjunto de programas del sistema operativo se le dan también los nombres de ejecutivo supervisor, director, rutina maestra. . . En otros casos, el nombre de supervisor o de programas de supervisión engloba a todo lo que aquí hemos lla-

mado programas de control, o sea los programas encargados de la gestión del sistema, de la gestión de trabajos y de la gestión de datos.

El objetivo de este programa es conseguir una utilización óptima de los recursos de la unidad central, coordinándolo con la gestión de los periféricos.

Es indispensable para la ejecución de todo tipo de programa, por lo que siempre está parcial o totalmente en memoria. En este último caso se le suele dar el nombre de ejecutivo residente, y entonces suele incluir, además de la gestión del sistema, la gestión de entradas/salidas, que aquí consideramos como capítulo aparte.

Las funciones a realizar dependerán de que el sistema operativo esté orientado a monoprogramación o multiprogramación. Podríamos citar las siguientes:

A) **En monoprogramación**:

- Tratamiento de errores de memoria. Estos se pueden detectar por un error de paridad que activa un indicador determinado y que al detectarlo el procesador cede control a la rutina correspondiente. Esta averiguaría si es un error real del hardware o un error por transmisión defectuosa, en cuyo caso trataría de corregir la situación repitiendo la transmisión.
- Tratamiento de errores de programa. Se produce por códigos de operación irreconocibles o cuando las instrucciones no se ajustan a las condiciones establecidas. Cuando esto sucede se cede el control a la rutina correspondiente, que sacará el oportuno mensaje al operador.
- Realización de instrucciones por software. En el programa objeto se pueden incluir instrucciones que no se puedan realizar por hardware (no existen los circuitos necesarios en el modelo en que se está pasando el programa). Al detectarlas se cede el control a una rutina que ejecutaría esta instrucción.
- Control de interrupciones. Cuando se produce alguna, debido, por ejemplo, a alguna de las causas anteriores, se guarda el estado de indicadores, contenido del registro de control de secuencia, etc., para después poder continuar el proceso.
- Conseguir la simultaneidad entre proceso y operaciones de entrada/salida.
- Servir de enlace con los demás programas del sistema operativo.

B) **En multiprogramación**:

- Las de monoprogramación para cada uno de los programas que haya en memoria.
- Control de todo tipo de interrupciones (programas periféricos y terminales).
- Control de los programas existentes en memoria y de la asignación de memoria a los diferentes programas que van entrando según las necesidades. Reparte, en el tiempo, el acceso de los programas a los distintos órganos de tratamiento y a las unidades periféricas.

- Supervisión de tareas. Determina prioridades en la ejecución de tareas pendientes, asigna turnos de espera, provoca interrupciones por tiempo asignado a cada tarea, etc.
- Control de rutinas no residentes en memoria y que las necesita algún programa.

#### 7.3.2.2. Gestión de datos

Se le denomina también IOCS y ejecutivo de entrada/salida, aunque en general "gestión de datos" es un término más amplio, y se aplica a los sistemas operativos más evolucionados.

Con este conjunto de rutinas se localizan, identifican y transmiten los datos, al mismo tiempo que se supervisan y sincronizan las operaciones de entrada/salida. Según esto, las funciones serían:

A) **En monoprogramación:**
- Controla los canales y las unidades de entrada/salida. Inicia las operaciones de E/S y si encuentra una unidad ocupada coloca una anotación en la lista de espera.
- Controla la transmisión de los registros físicos o bloques. Así, por ejemplo cuando un canal provoca una interrupción por haber terminado una operación de E/S. averigua cuál es la unidad causante y cede el control a la rutina de verificación correspondiente. Si detecta error, puede ceder control a otra rutina que intente corregirlo, etcétera.
- Controla la manipulación de los registros lógicos dentro del bloque. De esta forma se consigue que el programador opere siempre con estos registros lógicos en sus instrucciones de E/S.
- Rutinas para abrir y cerrar archivos, o sea para comenzar y finalizar el tratamiento de los archivos, controlando la manipulación errónea o no autorizada de determinados ficheros, gestiones de espacio, etc.
- Rutinas para facilitar el acceso de acuerdo con las distintas organizaciones de ficheros y de métodos de transferencia de datos.
- Llamada a rutinas diversas.

B) **En multiprogramación:**
- Esencialmente son las mismas que hemos indicado en monoprogramación, sólo que ahora con una mayor complejidad debido a la existencia de varios programas a los que se va cediendo control sucesivamente, y programas que, en una multiprogramación avanzada, pueden tener acceso a los mismos archivos.

#### 7.3.2.3. Gestión de trabajos (Monitor)

Es el programa o conjunto de programas que se encarga de leer, interpretar, iniciar y terminar los distintos trabajos que constituyen una explotación. Por tanto, es el encargado de la automatización en el paso de un programa a otro de una cadena, enviando los mensajes oportunos al operador.

En algunas ocasiones se hace distinción entre lo que es el **cargador** (el encargado de llevar a memoria, a sus direcciones adecuadas, los distintos programas que se van a ejecutar) y el **monitor**, que es el que llama al cargador. En otros sistemas el colaborador del monitor, con misión análoga al cargador, pero funcionamiento distinto, se le llama **editor** (linkage editor).

Con todo lo dicho ya se han indicado las funciones a realizar por los programas de gestión de trabajos, pero se podría hacer una mayor sistematización.

A) **En monoprogramación**:
- Lee, interpreta, inicia y termina los distintos trabajos.
- Efectúa el encadenamiento de los programas de acuerdo con unas **instrucciones de control.** Son instrucciones en un lenguaje que normalmente se llama lenguaje de control, y que sirven para indicar al monitor lo que ha de hacer.
  También se llaman **fichas de control**, pues en principio siempre se utilizaban fichas para estas instrucciones. Gracias a esto se modifica la secuencia de ejecución de programas.
- Sirve de comunicación entre el equipo y el operador.
- Llamada a rutinas diversas.

B) **En multiprogramación**:
- Las mismas funciones ya indicadas, pero atendiendo a los distintos programas existentes.
- Programación de los turnos de espera, debiendo elegir del trabajo a iniciar según el orden de prioridad atribuido en las fichas de control.
- Almacenamiento de resultados en memoria externa para su posterior impresión.

### 7.3.3. Programas de servicio

#### 7.3.3.1. Traductores

Son programas (conjuntos de programas) que tienen como datos de entrada un programa escrito en un lenguaje determinado y como salida el mismo programa en otro lenguaje.

En general, a los programas que manejan como datos otros programas se les llama **metaprogramas.**

El caso más frecuente de traducción es el de un programa en un determinado lenguaje de programación (programa fuente) a un programa en lenguaje de máquina (programa objeto). Los traductores se estudiaron en detalle en el Capítulo 5.

#### 7.3.3.2. Rutinas de utilidad

Llamados también rutinas de servicio. Incluye una serie de programas que realizan funciones necesarias, o al menos "útiles", para el sistema o el usuario y que, al ir incluidos en el sistema operativo, no es necesaria su programación cuando vayan a utilizarse.

Podemos distinguir dos tipos. Los programas para manipulación de datos y para servicios del sistema.

A) **Manipulación de datos:** Son todos aquellos que trabajan con ficheros, y podríamos citar:
   - Programas de intercalación.
   - Programas de clasificación.
   - Programas de transposición de soporte.
   - Programas de transcodificación.
   - Programas de construcción, reorganización y utilización de ficheros.
   - Programas diversos.

B) **Servicios del sistema:** Los que realizan funciones del sistema, entre los que podríamos citar:
   - Programas para manejo de librerías.
   - Programas para creación de calendarios, a efectos de trabajo, protección de archivos, etc.
   - Programas para utilización de archivos índices (registro de trabajos, errores e incidencias).
   - Programas para depuración de programas o pruebas de programas.
   - Pruebas del sistema para probar el correcto funcionamiento de las distintas unidades o para localizar errores (en mantenimiento).
   - Programas diversos.

#### 7.3.3.3. Programas de aplicación

Son todos los programas del usuario que no se incluyen en ninguno de los tipos anteriores. Se pueden distinguir dos tipos:

A) **Programas escritos por el usuario,** del tipo que sea.

B) **Packages de aplicación**: son programas completos que solucionan problemas comunes a distintos usuarios con modificaciones mínimas. Tienen el inconveniente de que son programas standard y pueden no adaptarse totalmente a nuestras necesidades, pero la gran ventaja de poder hacer su utilización inmediata y a un precio más bajo de lo que le costaría desarrollarlo al propio usuario. Por esto están adquiriendo cada vez más importancia.

# CAPITULO 8. PROCESO DE MECANIZACION DE UN TRABAJO

## 8.1. Fases de la mecanización

Hasta el momento se ha presentado la estructura básica de un ordenador, su funcionamiento interno y las herramientas del software más comunes para su utilización. Se ha visto que para que el ordenador realice tareas que anteriormente se venían efectuando de una manera manual, el usuario debe confeccionar un programa que obedezca a la secuencia de trabajos que el ordenador debe realizar (programación del usuario).

La figura 1 presenta un esquema-resumen de las distintas fases por la que discurre el diseño y posterior explotación de un programa que confecciona el usuario de un ordenador para resolver un determinado problema.

El proceso comienza, para cada aplicación concreta, por la ejecución de un **Análisis de Oportunidad,** que permitirá decidir si el problema es o no mecanizable, a qué costo, qué posibles cambios implica en las estructuras de la empresa y el CPD, etc.

Tras decidir si es viable la mecanización del problema, se efectuará un **Análisis Funcional**, en que se especifica tanto la información de entrada que tendrá que proporcionarse al ordenador, como la información de salida, es decir, los resultados que se esperan del ordenador, especificándose el proceso a realizar para llegar a dichos resultados. Posteriormente, se pasa al **Análisis Orgánico** en el que se definen los procesos parciales que deberá ejecutar el ordenador, que se plasman en un **organigrama** (secuencia de los citados procesos). La siguiente fase consiste en escribir en un **lenguaje** determinado la secuencia de operaciones reflejada en el organigrama, fase que consiste en la **codificación** del programa a instrucciones y

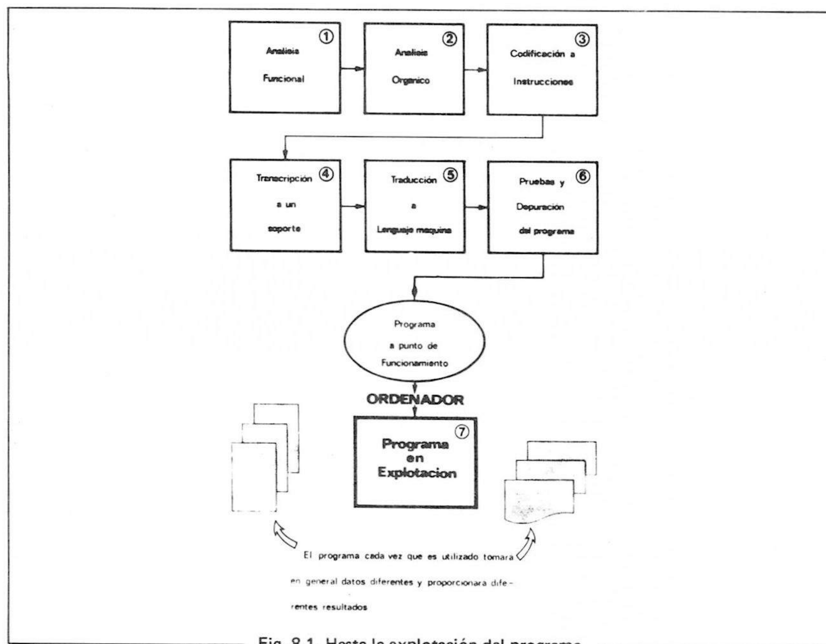



Fig. 8.1. Hasta la explotación del programa.

que normalmente se conoce simplemente como **programación.** A continuación se recuerda la necesidad de efectuar una transcripción del programa, (que en principio estaría materializado sobre un documento de papel), a un cierto soporte de información que pueda leer una unidad de entrada adecuada del ordenador. El programa así establecido se denomina programa fuente.

Una vez el programa en un soporte determinado, se procede a la lectura y **traducción** del citado programa fuente (compilación en caso de lenguaje de alto nivel).

Una vez el programa en lenguaje máquina (ceros y unos), no puede asegurarse todavía su corrección. Si bien el programa traductor comprobó que no existían incorrecciones de forma en el programa, nada se puede asegurar sobre si las instrucciones escritas por el programador obedecían o reflejaban la ejecución de las operaciones previstas; por ejemplo si el usuario pensaba efectuar la división de dos cantidades y para ello escribió la instrucción.

MULT A, B (Multiplicar A por B).

en lugar de

DIV A, B (Dividir A entre B).

Entonces el programa traductor no encontraría ningún error de lenguaje. Ahora bien, una vez traducido el programa, ejecutaría otras operaciones no previstas en el análisis, apareciendo lógicamente resultados erróneos.

Para evitar esto, los programas en lenguaje máquina (programas objeto) deben someterse a una **depuración** a base de una serie de pruebas (juego de ensayos) que prevean las situaciones, que en la práctica, van a producirse durante la ejecución del programa.

Una vez el programa en condición de ser utilizado se pone en funcionamiento o **explotación.** No acaba aquí, sin embargo, el proceso. Dado que el programa viene a realizar de una manera automática un determinado proceso, pudiera ocurrir que las condiciones de éste o su entorno varíen a lo largo del tiempo; un ejemplo puede ser un programa que efectúa el cálculo de la nómina de una empresa; sí, por ejemplo, varían las condiciones de tributación a la Seguridad Social, o bien la empresa implanta un nuevo sistema de primas, etc., el programa deberá modificarse para ajustarse a la nueva situación. Con ello se pretende poner de manifiesto la posibilidad de reciclar el proceso completo de programación, bien introduciendo pequeñas modificaciones del programa, bien teniendo que rediseñar, a nivel de análisis funcional, toda la aplicación(*)

---

*En el caso de una alteración muy significativa del entorno o bien de los procesos que dieron lugar a la confección del programa actualmente en explotación, puede ser más conveniente dar a éste definitivamente de baja y confeccionar otro, de acuerdo con las nuevas bases del problema planteado.

## 8.2. Análisis

### 8.2.1. Introducción

Es indispensable que el programa del usuario, ejecutado por el ordenador, efectúe todas y cada una de las tareas que configuraban la aplicación general a mecanizar, (frecuentemente al mecanizar un trabajo se incluyen, aprovechando las posibilidades del ordenador, nuevas tareas no realizadas anteriormente).

Para ello el usuario deberá analizar meticulosamente el problema que trata de resolver, sin dejar ningún detalle "colgado" ni lo más mínimo a la improvisación, pues no debe olvidarse que el ordenador ejecuta solamente aquello que ha sido programado previamente. Por lo tanto deberá observar con atención todos los procesos que se tratan de mecanizar con el ordenador y preparar la programación necesaria que responda a todas las posibilidades que puedan darse en el conjunto de la aplicación a resolver.

Al conjunto de este estudio del problema a mecanizar, se le denomina **análisis** de una aplicación. Normalmente se consideran tres fases:
- Análisis de oportunidad.
- Análisis funcional.
- Análisis orgánico.

Las dos últimas fases no tienen realmente una separación clara entre sí y va siendo una tendencia general el prescindir de su diferenciación. Puede indicarse simplemente que el análisis funcional está enfocado hacia cuál es el problema a resolver, sin tener en cuenta para nada las características del ordenador que en el futuro se va a emplear; por el contrario, el análisis orgánico constituye la fase más "cercana" al ordenador, que estudia como se utilizará éste. Entre una y otra fase, primera y última, hay una sucesión de pasos que acercan cada vez más la una a la otra. No obstante motivos didácticos pueden hacer aconsejable exponer separadamente las dos etapas (ver figura 2).

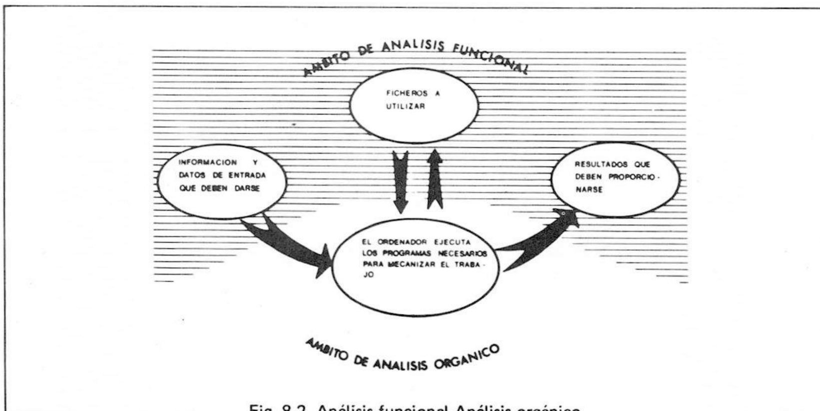


Fig. 8.2. Análisis funcional-Análisis orgánico.

### 8.2.2. Análisis de oportunidad

Es la primera fase, y su objeto es definir el costo de la mecanización de un problema dado, no sólo económicamente, sino también en tiempo, personal, etc, así como prever posibles cambios en la estructura administrativa y otros factores a tener en cuenta.

Resulta evidente que una estructura diseñada para un trabajo no mecanizado puede implicar una carga a la mecanización, ya que ciertos documentos, puestos de trabajo, etc, pueden no resultar operativos una vez automatizados los procesos. Este problema puede no aparecer de forma evidente si la mecanización se efectúa en pasos pequeños, programando aplicaciones aisladas. Si esto se efectúa así, es casi seguro que los circuitos de información en la empresa estarán, una vez mecanizada ésta, degenerados en mayor o menor grado.

El análisis de oportunidad pretende mantener una visión global, adaptando las estructuras dinámicamente, teniendo presente un objetivo final, la total mecanización. Hay que tener en cuenta que alcanzar este objetivo puede ser una labor de varios años.

En cualquier caso, este análisis ofrecerá datos que permitirán definir la conveniencia y costo de la mecanización, en cierta fecha, de una aplicación dada.

### 8.2.3. Análisis funcional

Ante el planteamiento de una mecanización, por ejemplo de cierto trabajo en un departamento de una empresa, los analistas proceden en primer lugar a estudiar la situación actual del trabajo en cuestión. Así por ejemplo:

- Cada una de las secuencias parciales del trabajo (subtrabajos).
- Documentos que recibe, genera, almacena y proporciona a su salida dicho trabajo (albaranes, facturas, órdenes de envío, etc.), y circuitos por los que éstos circulan.
- El volumen y características de la información que tratan.
- Ficheros de datos que se consultan o actualizan (clientes, artículos, etc.).
- Etc. etc.

Tras este planteamiento previo los analistas decidirán las acciones más oportunas a tomar, en base a:

- Mecanizar todo el trabajo.
- Racionalizar al máximo este trabajo; reducir la información circulante al mínimo imprescindible, evitar redundancias en los documentos, eliminar subtareas antieconómicas. (En definitiva optimizar el proceso).

A la vista de los planteamientos previos, los analistas proceden a configurar una solución para resolver la mecanización del trabajo. Para ello:

- Se definirán **los documentos** necesarios de **entrada** (formato de albaranes, listas diversas, etc.).
- Se definirán los documentos que han de proporcionarse como **salida** (resultados, cartas de envío, recibos, etc.).
- Se definirán los tipos definitivos de datos a tratar y los **canales de circulación** de los datos de entrada y de salida a través de la empresa (o dependencia).
- Se definirá el medio o medios de **soportes** de la información tratada (transcripción de datos de entrada a cassette, tarjetas perforadas, etc.).
- Se definirán los **ficheros** que se utilizarán (tipos de datos que han de almacenarse, cantidad, configuración de ficheros, soportes, etc.).
- Se determinarán **plazos de tiempo**, responsables de ejecución de trabajos, etc.

La figura 3 presenta un esquema en que se muestran las ideas arriba enumeradas con respecto al análisis funcional, en el caso concreto de la mecanización de un proceso de facturación.

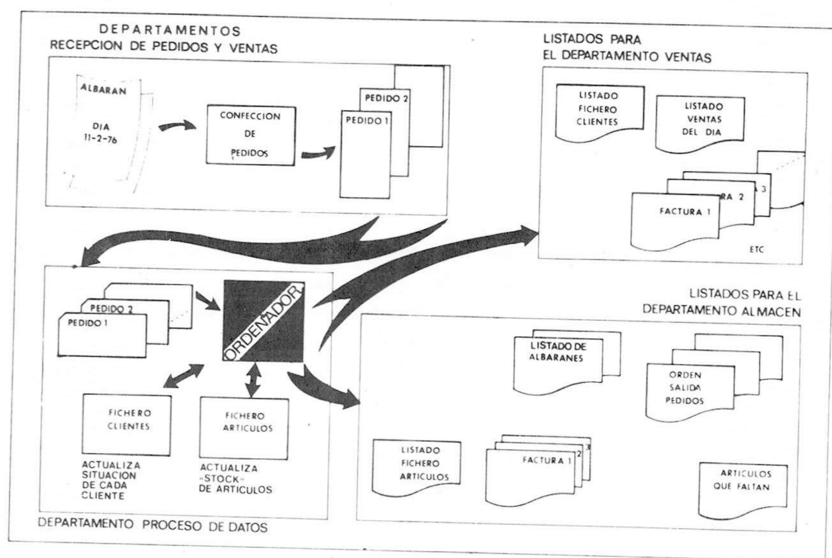



Fig. 8.3. El análisis funcional estudia la información de entrada, la información procesada por el ordenador y la de salida proporcionada. Ejemplo simplificado de facturación.

### 8.2.4. Análisis orgánico

#### 8.2.4.1. Concepto

La última fase del análisis de una aplicación, estudia **los procesos** que ha de efectuar el ordenador para, mediante la información de entrada y los ficheros de datos que se especifican en la fase de análisis funcional, proporcionar los resultados de salida igualmente especificados en dicha fase. Por lo tanto, no queda sino centrarse en el trabajo que ha de ejecutar el ordenador.

Existen diversos procedimientos para preparar y especificar la secuencia de operaciones que debe efectuar el ordenador. Uno de ellos, por ahora el más habitual, es la utilización de organigramas o diagramas de flujo.

Además de los organigramas que explican detalladamente el proceso general de la aplicación, el analista deberá especificar:

- Características detalladas de la disposición de los datos en los soportes de entrada (tipo y tamaño de los datos de entrada en tarjetas, cassettes, etc.).
- Características detalladas de los datos que existen en los ficheros (longitud y situación de cada uno, tipo de agrupamiento u organización, manera de tratarlos, etc.).
- Características detalladas de la disposición de datos en la documentación de salida a fin de preparar los procesos necesarios para obtenerlos en la manera prevista.
- Tipo y característica de la periferia que utilice.
- Disponibilidad de memoria; etc.

Organigramas y Tablas de Decisión son los dos instrumentos de que dispone el analista para especificar el proceso, y serán la base del trabajo del programador, que apoyándose en ellos detallará las instrucciones que el ordenador deberá ejecutar para efectuar las acciones especificadas en ellos.

#### 8.2.4.2. Organigramas

Como se dijo anteriormente, un organigrama es la representación gráfica de un proceso algorítmico orientado al ordenador. Un organigrama se construye uniendo mediante flechas que indican el flujo lógico del proceso, esto es, la secuencia de ejecución, una serie de símbolos, cada uno de los cuales representa una acción a tomar por el ordenador (decisión, lectura de datos, escritura, comienzo y fin de proceso, etc.).

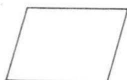
Dentro de los símbolos se detalla la acción concreta a efectuar.

En general existen dos niveles en que se utiliza el organigrama:

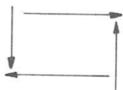
– A nivel funcional de la aplicación; en cada casilla se especifican las líneas generales de los procesos a efectuar.
– A nivel orgánico; se representan con mucho más detalle los datos o variables que intervienen en el proceso y operaciones que se efectúan con ellos(*). Este es el último paso antes de comenzar la transcripción a instrucciones del programa, y puede ser efectuado por el programador o proporcionado por el analista, dependiendo quizá de la complejidad del proceso.

Los símbolos más comúnmente utilizados en los organigramas son los siguientes:

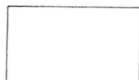
**Símbolos básicos**



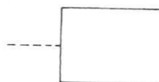
**Entrada/Salida**
Representa la Entrada o Salida de información en o desde cualquier dispositivo.



**Flujo**
Especifica la dirección de flujo o secuencia de proceso.



**Proceso**
Representa la ejecución de cierta operación o grupo de operaciones.

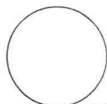


**Comentario**
Se utiliza para escribir notas aclaratorias sobre algún aspecto del proceso.

**Símbolos de Entrada/Salida específica**



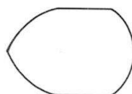
**Tarjeta perforada.** Lectura o perforación.



**Cinta magnética.** Lectura o grabación.

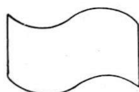


**Entrada manual.** Introducción de datos por teclado, activación o desactivación de switches, etc.



**Visualización.** Salida por pantallas, visualizadores, luces, etc, de información destinada al operador.

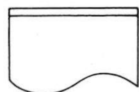
*A este segundo tipo de organigrama se le conoce frecuentemente con el nombre de ordinograma.



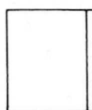
**Cinta perforada.** Lectura o perforación.



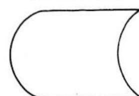
**Impresión.** Salida de información por cualquier dispositivo impresor.



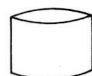
**Impresora de líneas.** Cuando coexisten impresoras seriales y de líneas, el símbolo anterior se utiliza para las seriales y éste para las de líneas.



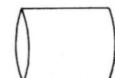
**Ficha de banda magnética.** Lectura o grabación.



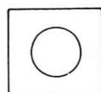
**Discos o tambores magnéticos.** Lectura o grabación.



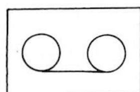
**Discos magnéticos** específicamente.



**Tambores magnéticos** específicamente.

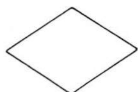


**Diskettes** o discos flexibles. Lectura o grabación.

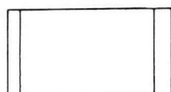


**Cassettes.** Lectura o grabación.

### Símbolos de proceso específico



**Decisión.** Representa un punto del proceso desde el que pueden tomarse varios caminos, dependiendo de ciertas condiciones.



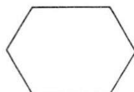
**Rutinas.** Representa la ejecución de un grupo de operaciones especificado en otro lugar.



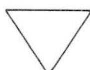
**Operación manual.** Representa la ejecución por el operador de algún tipo de proceso que requiere el programa.



**Operación auxiliar.** Representa la ejecución por el operador de alguna operación en equipos fuera del control del ordenador.



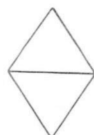
**Disposición.** Representa la modificación de alguna o algunas instrucciones del programa.



**Fusion.** De varios conjuntos de datos en uno sólo.

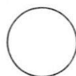


**Extracción.** Obtención de varios conjuntos de datos a partir de uno sólo.

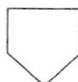


**Clasificación** (SORT). Ordenación de un conjunto de datos con ciertos criterios.

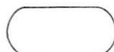
**Símbolos adicionales.**



**Conector interno.** Representa el salto de un punto a otro del organigrama.



**Conector externo.** Representa el salto de un punto del organigrama a otro punto del organigrama que no figura en la misma página.



**Terminal.** Representa el inicio o fin del proceso.

La figura muestra el organigrama del proceso de facturación visto en la figura 8.3.

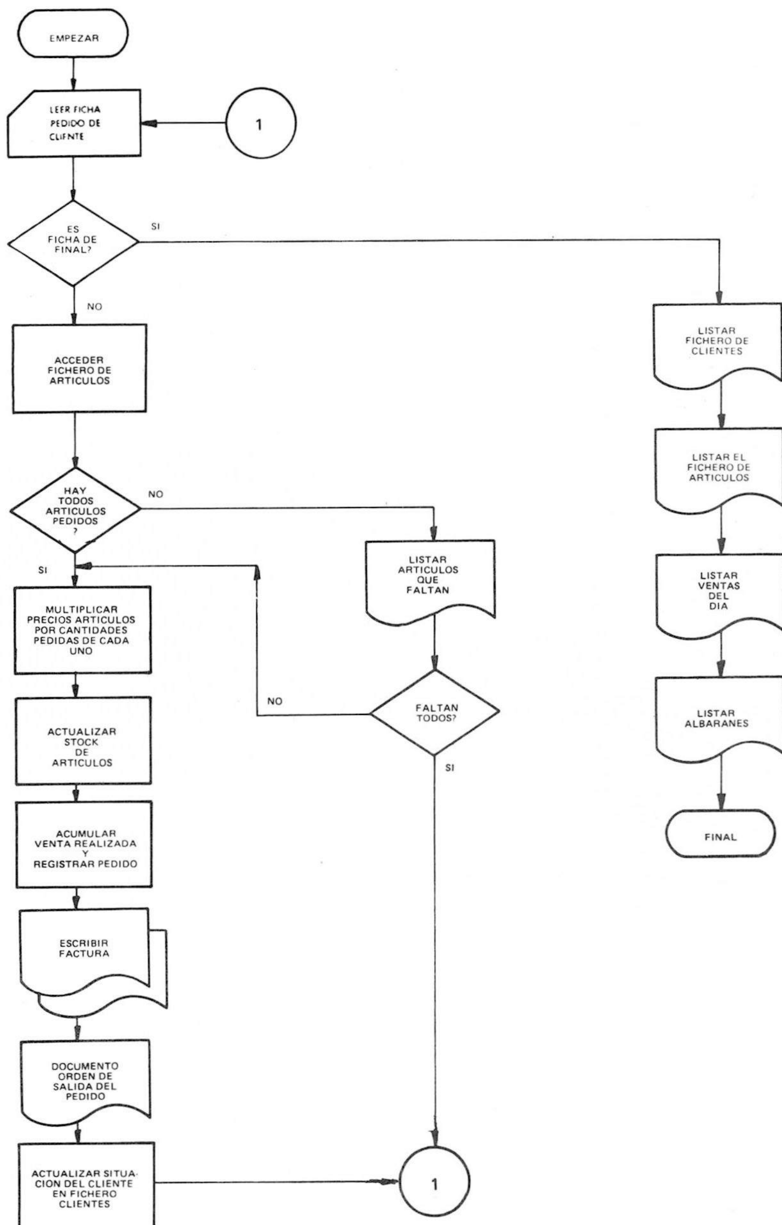



Fig. 8.4. Organigrama simplificado de facturación.

### 8.2.4.3. Tablas de decisión

Las tablas de decisión son representaciones gráficas de las acciones a tomar en ciertas condiciones. En tal sentido, se utilizan sustituyendo a los diagramas de flujo (organigramas) cuando las condiciones a establecer son complejas. La diferencia principal entre unos y otros es que una Tabla de Decisión no indica explícitamente el procedimiento a utilizar para efectuar las acciones resultantes.

Una tabla de decisión se estructura, tal y como aparece en la fig. 5, formando cuatro cuadrantes por medio de dos líneas de doble trazo.

Reglas

Condiciones

Entradas de condiciones

Acciones

Entradas de acciones

Fig. 8.5. Estructura de una tabla de decisión.

El cuadrante superior izquierdo especifica las condiciones a investigar, detallándose en el cuadrante superior derecho los posibles resultados de dicha investigación, en una serie de columnas (Reglas), cada una de las cuales especificará una configuración de resultados para cada condición.

La mitad inferior de la tabla se refiere a las acciones a tomar; las posibles acciones se detallan en el cuadrante izquierdo, especificándose en el cuadrante derecho la configuración de acciones a tomar para cada grupo de resultados de las condiciones (para cada Regla).

La figura 6 muestra un ejemplo de condición compleja en un programa, en forma de organigrama; dependiendo de los valores de las variables X, Y, Z, se ejecutará una de las acciones A1 a A6. La tabla de decisión que representa el mismo proceso aparece en la figura 7. En este ejemplo se observan dos de las características principales de una tabla de decisión; primero, que cada regla es independiente de las demás, es decir, el orden en que aparece en la tabla no es indicativo, ya que la regla se cumplirá si se cumplen las condiciones en ella especificadas, independientemente de las otras reglas, y segundo, que mientras en un organigrama es relativa-

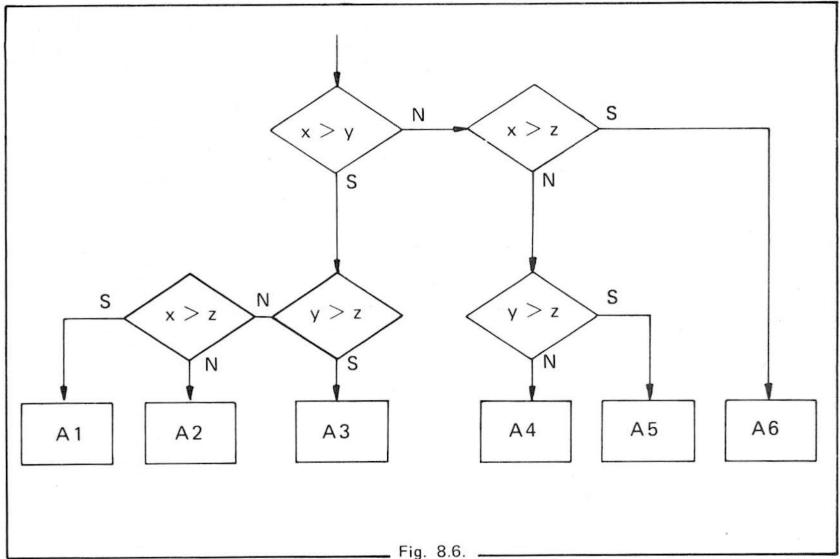



Fig. 8.6.

| | R1 | R2 | R3 | R4 | R5 | R6 |
|---|---|---|---|---|---|---|
| x>y | S | S | S | N | N | N |
| x>z | S | N | – | N | N | S |
| y>z | N | N | S | N | S | – |
| Ejecutar | A1 | A2 | A3 | A4 | A5 | A6 |

Fig. 8.7.

mente sencillo olvidar especificar cierta condición, al construir una tabla se **barren** todas las configuraciones posibles, por lo cual la posibilidad anterior es prácticamente nula.

En la sección de Entrada de Condiciones de la tabla aparecen signos S, N y –; S indica que se cumple la condición, N que no se cumple y – que es indiferente que se cumpla o no la condición especificada.

Acciones y Condiciones pueden ponerse en dos formatos:

–Entrada limitada
–Entrada extendida

cabiendo además la posibilidad de formar entradas mixtas, como mezcla de las dos anteriores.

Las condiciones de la tabla de la figura 7 se presentan en formato de entrada limitada, ya que la condición se especifica totalmente en el cuadrante de Condiciones, apareciendo como entrada únicamente S o N para especificar si se cumple o no. Por el contrario, la sección de Acciones sólo especifica **Ejecutar**, completándose con la información que aparece en las entradas (Ejecutar A1, A2, etc.), es decir, está en formato de entrada extendida.

El ejemplo de la figura 8 muestra Condiciones en formato de entrada extendida y Acciones en formato de entrada limitada. La regla X abarcará todas las combinaciones de condiciones no especificadas en las otras reglas.

| Cantidad compra > | 5 | 10 | 50 | 10 | 50 | 100 | 10 | 100 | X |
|---|---|---|---|---|---|---|---|---|---|
| Tipo cliente | A | A | A | B | B | B | C | C | |
| Aplicar 10 % descuento | X | | | X | | | X | | |
| Aplicar 20 % descuento | | X | | | X | | | X | |
| Aplicar 30 % descuento | | | X | | | X | | | |
| No aplicar dto. | | | | | | | | | X |
| Aplicar 3 % p. pago | | X | X | | X | X | | X | |
| No aplicar p. pago | | | | | | | | | X |

Fig. 8.8.

## 8.3. Programación

### 8.3.1. Fases

Una vez analizado totalmente el problema, confeccionados los organigramas, diseñados los ficheros, etc, se comienza la fase de programación, esto es, la conversión a instrucciones de los procesos especificados en el análisis.

Para aplicaciones de gestión administrativa, la programación es sin duda la fase más laboriosa y costosa en tiempo.

Dependiendo del detalle al que se haya descendido en el análisis, la programación será un trabajo de simple conversión a instrucciones o requerirá del programador la ejecución de organigramas más detallados como paso previo.

La fase de programación constará de dos pasos, la **codificación** o escritura de las instrucciones y la **depuración** o prueba del programa a fin de detectar errores en la fase anterior. Una vez finalizada esta fase, el programa estará listo para ser ejecutado.

### 8.3.2. Codificación

La codificación se realizará en el lenguaje que indique el análisis, dependiendo del tipo de aplicación de que se trate. En cualquier caso, de un mismo análisis y en un mismo lenguaje pueden obtenerse muchos programas diferentes, aunque todos realicen el mismo trabajo; la diferencia en el orden de las preguntas, evitación de pasos inútiles en ciertas condiciones, establecimiento de rutinas para los pasos que se presentan en varios puntos del proceso, condicionarán la bondad del resultado en cuanto a rapidez de ejecución y ocupación de memoria, que son los puntos calificados de un programa.

Por supuesto, la codificación vendrá condicionada por la estructura del lenguaje, dependiendo de la cual el programador podrá utilizar en su programa estructura de procedimientos, bloques, técnicas de recursividad y otros elementos que simplifiquen su trabajo u optimicen sus resultados.

Las figuras 10 y siguientes muestran la codificación en diversos lenguajes del proceso de ordenación creciente de 1.000 números dados, cuyo organigrama se representa en la figura 9.

Se supondrá que los números estarán en memoria al comenzar el proceso. En el ejemplo en Ensamblador, la dirección del 1º de ellos tomará el nombre simbólico NUM, direccionando cada elemento a base de sumar a NUM el valor de un desplazamiento llamado D1 o D2. En los otros lenguajes, los números formarán una matriz unidimensional llamada NUM.

El método de ordenación será el siguiente: se compara cada número con el que le sigue; si el 1º es más pequeño o igual, se dejan tal como están, sino, se intercambian. Cada vez que se hace un cambio se incrementa un contador CONT. Cuando se llegue al último número, y si ha habido cambios, se vuelve a comenzar por el primero. Cuando se comparen los 1000 números sin efectuar cambios será que están todos en orden, en cuyo momento se finaliza el proceso.

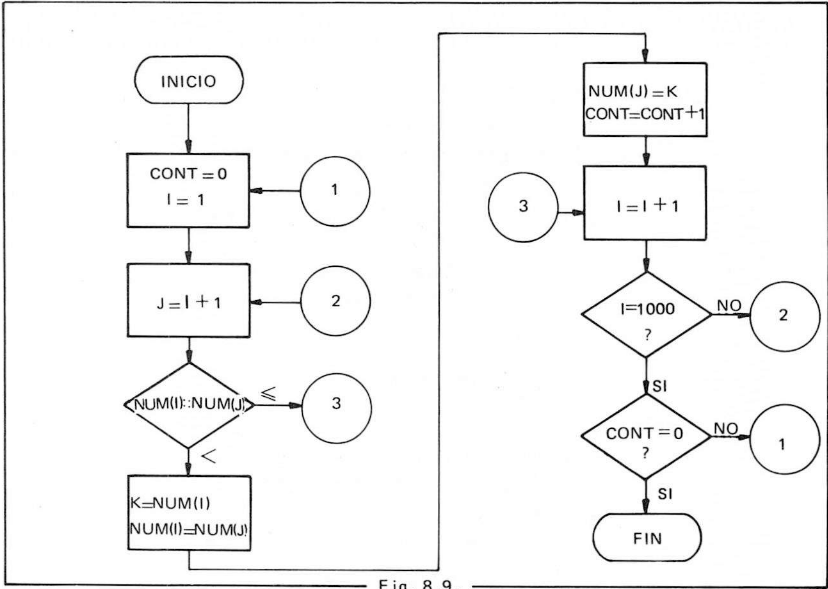



Fig. 8.9.

**Ensamblador ideal**

Instrucciones:

- LMP – Limpiar
- CAR – Cargar el acumulador con el contenido de un cierto campo o con un valor.
- TR – Transportar a cierto campo el contenido del acumulador.
- SUM – Sumar el contenido del acumulador al de cierto campo, quedando el resultado en el acumulador
- RES – Restar del contenido del acumulador el contenido de cierto campo quedando el resultado en el acumulador
- BN – Bifurcar a cierta instrucción si el contenido del acumulador es negativo
- BDC – Bifurcar a cierta instrucción si el contenido del acumulador es distinto de cero.
- FIN – Fin de programa.

| Nº de instrucción | Código de operación | Operando |
|---|---|---|
| 1 | LMP | CONT |
| 2 | LMP | D1 |
| 3 | CAR | '1' |
| 4 | SUM | D1 |
| 5 | TR | D2 |
| 6 | CAR | NUM+D1 |
| 7 | RES | NUM+D2 |
| 8 | BN | 18 |
| 9 | CAR | NUM+D1 |
| 10 | TR | K |
| 11 | CAR | NUM+D2 |
| 12 | TR | NUM+D1 |
| 13 | CAR | K |
| 14 | TR | NUM+D2 |
| 15 | CAR | '1' |
| 16 | SUM | CONT |
| 17 | TR | CONT |
| 18 | CAR | '1' |
| 19 | SUM | D1 |
| 20 | TR | D1 |
| 21 | CAR | '1ØØØ' |
| 22 | RES | D1 |
| 23 | BDC | 3 |
| 24 | CAR | CONT |
| 25 | BDC | 1 |
| 26 | FIN | |

Fig. 8.10.

**COBOL**

```
INICIO
        MOVE ZEROS TO CONT.
        MOVE 1 TO I.
        MOVE 2 TO J.
        PERFORM CICLO THRU INCRE 1ØØØ TIMES.
        IF CONT IS EQUAL TO ZERO GOTO FIN ELSE GOTO INICIO.
CICLO.
        IF NUM(J) IS NOT GREATER THAN NUM(I) GO TO INCRE.
        MOVE NUM(I) TO K
        MOVE NUM(J) TO NUM(I)
        MOVE K TO NUM(J)
        ADD 1 TO CONT.
INCRE.
        ADD 1 TO I.
        ADD 1 TO J.
FIN.
        STOP RUN.
```

Fig. 8.11.

```
BASIC

1   SW = Ø
    FOR I = 1 TO 999
              IF NUME (I+1) < NUME (I) THEN
                        K = NUME (I)
                        NUME(I) = NUME(I+1)
                        NUME(I+1) = K
                        SW = SW + 1
                        END
    NEXT I
    IF SW ≠ Ø THEN GOTO 1
    END
```

Fig. 8.12.

```
FORTRAN

  1 | CONT = Ø
    | DO 9   I = 1,999,1
    | J = I + 1
    | IF(NUM(J) ≥ NUM (I)) GO TO 9
    | K = NUM(I)
    | NUM(I) = NUM(J)
    | NUM(J) = K
    | CONT = CONT + 1
  9 | CONTINUE
    | IF (CONT ≠ Ø ) GO TO 1
    | STOP
    | END
```

Fig. 8.13.

### 8.3.3. Depuración

Es la fase de prueba de los programas. Introduciendo éste en máquina, el traductor (ensamblador, compilador, etc) proporcionará los errores de transcripción y sintaxis que se hayan producido. Se corrigen éstos y se vuelve a introducir el programa hasta que no ofrezca errores de este tipo. En este punto el programa ya es ejecutado por el ordenador, para lo cual el programador habrá preparado un **juego de ensayo** (un conjunto de datos de entrada que cubra todos las posibilidades). De esta forma, al examinar los resultados de salida se podrán detectar los errores lógicos del programa, volviendolo a ejecutar ya corregido hasta eliminar todos los errores.

En cualquier caso, en programas medianamente complejos es prácticamente imposible prever en el juego de ensayo todas las posibilidades, por lo cual algunos errores se detectarán únicamente cuando, ya en explotación, se den los casos extraños que el programador no previó.

Los Sistemas Operativos suelen incluir facilidades de depuración **(debugging)** para ayudar al programador en la fase de detección de errores, que es sin duda la más compleja de todas las de la mecanización.

## 8.4. Explotación

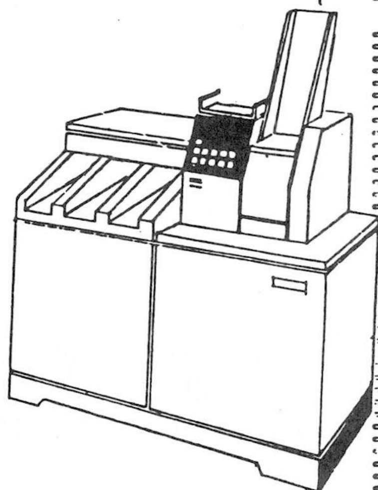
Es la fase en que el programa ejecuta su trabajo normalmente, ya con datos reales. Como antes se dijo, pueden ir surgiendo eventuales errores no detectados en la depuración.

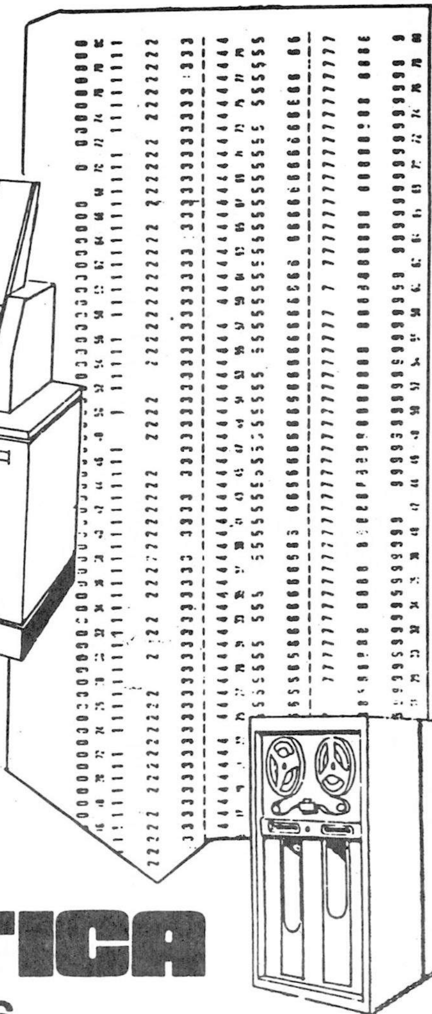
Una vez el programa lleva un tiempo en esta fase puede ser práctico tomar tiempos y comprobar si las ventajas supuestas en el análisis de oportunidad se establecen realmente.



Foto. 8.1. Explotación



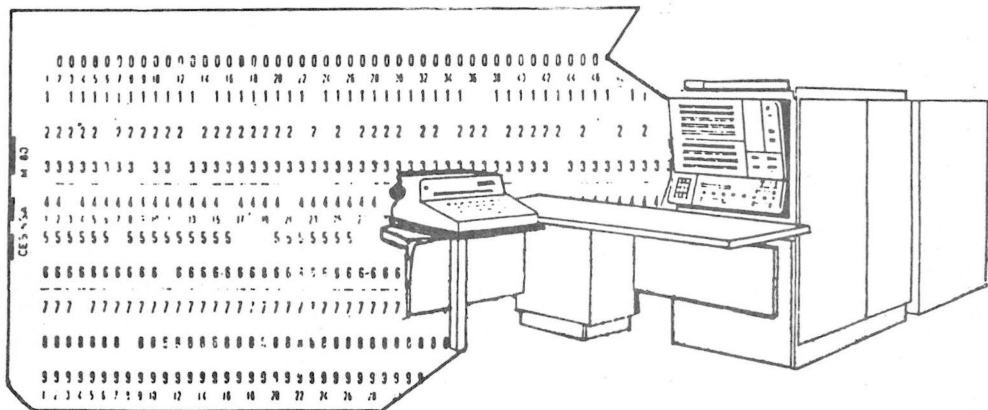




# INICIACION A LA

# NFORMATICA

## ORGANIGRAMAS



# O R G A N I G R A M A S

ESTUDIO TECNICO DE INFORMATICA



Depósito legal: M- 2393- 74. Madrid, enero de 1974.

DURANTE MUCHO TIEMPO HEMOS TRABAJADO EN LA PREPARACION DE ESTE LIBRO, PARA PODER AHORA, PRESENTARTE ESTA COLECCION DE PROBLEMAS, TODOS ELLOS RESUELTOS, QUE SUPONEMOS DE GRAN INTERES PARA TI, Y QUE DEBIDO A LA FORMA ORDENADA, CLARA Y CONCISA EN QUE HAN SIDO RESUELTOS, SERAN SIN DUDA UNA AYUDA INDISPENSABLE Y EFICAZ PARA LA COMPRENSION TOTAL DE ESTE METODO DE DESCRIPCION DE PROCESOS QUE SON LOS ORGANIGRAMAS.

AQUI TIENES NUESTRA COLABORACION.

AGRADECIENDOTE DE ANTEMANO LA BUENA ACOGIDA QUE DISPENSARAS A NUESTRO TRABAJO Y NO DUDANDO, QUE SABRAS BIEN DISCULPAR TODOS AQUELLOS ERRORES QUE PUDIERAS ENCONTRAR,TE SALUDA ATENTAMENTE

ESTUDIO TECNICO DE INFORMATICA

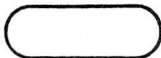
# INTRODUCCION

Un organigrama es una forma simple y esquemática de mostrar la lógica de un problema, es decir, los distintos pasos a seguir para obtener los resultados deseados.

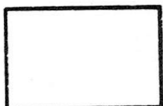
Hacer el organigrama en un problema sencillo no tiene mucha útilidad, pero si el problema es relativamente complejo, es recomendable hacer previamente un organigrama, ya que facilita no solo su posterior escritura, sino la comprensión de su forma de operar con miras a un posible útilizador del programa, además de facilitar en el futuro la introducción de posibles correcciones.

El organigrama puede mostrar la lógica del problema a grandes pasos o de forma muy detallada. Los organigramas más útiles, en general, son los de tipo intermedio.

Los simbolos y las técnicas usadas en organigramas son muy variadas. En la presente publicación se han útiñizado los simbolos más comúnes y de uso más generalizado. Estos son los siguientes :



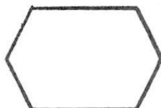
TERMINAL.- Puede indicar comienzo, fin, espera ó interrupción, así como salida de una subrutina cerrada.



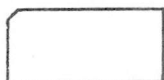
PROCESO.- Cualquier función de proceso u operación.



DECISION.- Decisión que determinará un camino a seguir entre varios(2 ó 3) alternativos.



INICIALIZACION.- Asignación de valores iniciales a determinados datos del proceso.



TARJETA PERFORADA.- Cualquier operación de entrada y salida mediante tarjeta perforada.



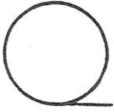
IMPRESION.- Cualquier operación de salida mediánte papel impreso.



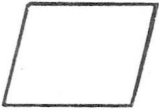
INDICA QUE POR LA PANTALLA VA A SALIR UN MENSAJE



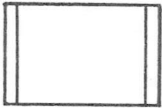
Introduce datos del Teclado



CINTA MAGNETICA.- Cualquier operación de entrada/salida en cinta magnética.



ENTRADA/SALIDA.- Cualquier operación general de entrada/salida.



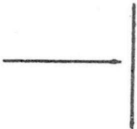
SUBRUTINA.- Indica subrutina o proceso predefinido



CONECTOR.- Permite enlazar un punto del organigrama con cualquier otro punto del mismo.



CONECTOR DE PAGINAS.- Permite enlazar un punto del organigrama con otro punto de otra página.



LINEAS DE FLUJO.- Permiten enlazar unas operaciones con otras, indicando la secuencia de ejecución de las mismas.



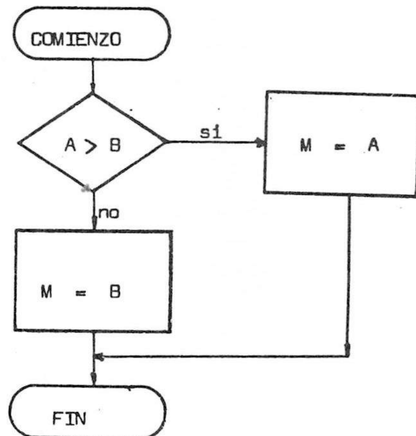
PUNTAS DE FLECHA.- Permiten indicar el sentido de recorrido de las lineas de flujo.

PROBLEMA **1**

Dados dos números A y B en memoria, hacer un organigrama para almacenar el mayor en M.

RESOLUCION :

La solución es inmediata, pues la sola comparación de ambos números determinará cual es el mayor :




En esta solución vemos que si los números A y B son iguales, se toma como mayor el B, que en este caso seria igual al A.

## PROBLEMA 2

Dados 6 números en memoria, hacer un organigrama para almacenar el menor en M.

### RESOLUCION

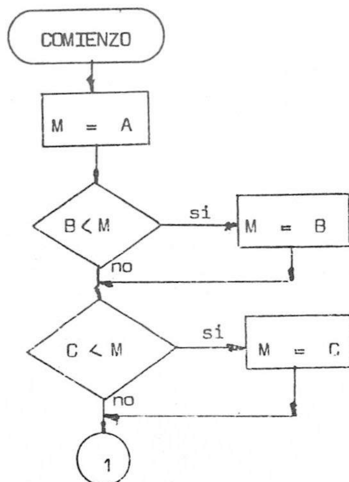
El proceso que vamos a describir consiste en suponer que el menor es el primero, almacenandolo pues en M. Para comprobarlo comparamos M (que suponemos que guarda el menor) con todos los demás. Si encontramos en alguna de estas comparaciones un número menor que el M, lo sustitui mos por el que guarda M.

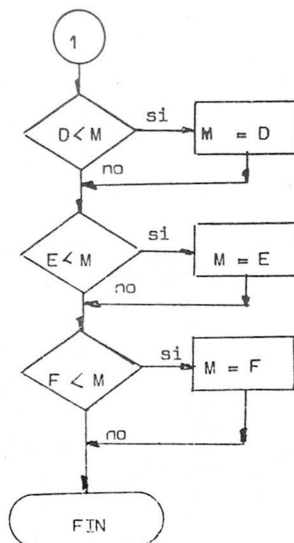
Al final en M tendremos efectivamente el menor.

El lector podrá comprobar con un ejemplo el funcionamiento del proceso.

Veamos primero el organigrama lineal.

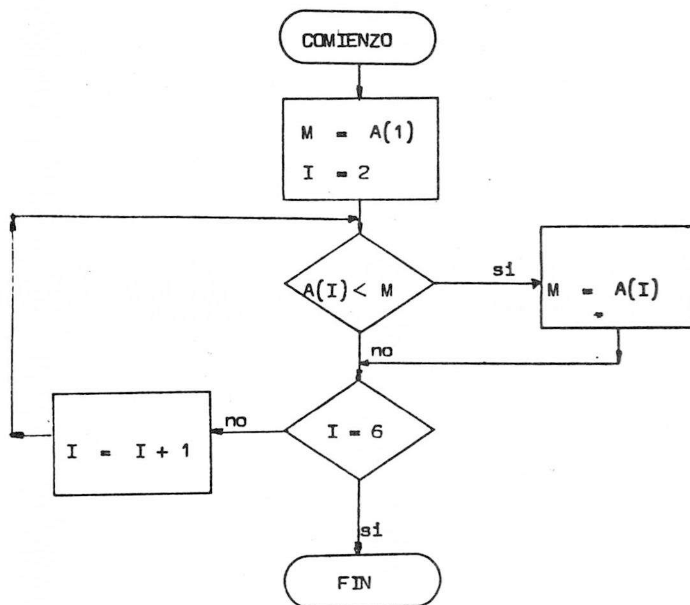
Supongamos que los 6 números están guardados en A, B, C, D, E y F.



El organigrama cíclico seria :

Suponemos que los 6 números forman un conjunto $A_i$.




Podemos observar que es más sencilla la solución lineal, pero que es más breve la cíclica, sobre todo en el caso más normal en que el número de elementos sea grande.

## PROBLEMA 3

### ORGANIGRAMA Nº 1

Se quiere multiplicar dos números, sacando el resultado por impresora. Cada número entra en una ficha distinta. Analizar el problema, hacer un organigrama para resolverlo.

### ORGANIGRAMA Nº 2

Analizar, hacer el ordinograma y probarlo, para un programa que liste en una impresora el contenido de dos fichas, en una línea.

### ORGANIGRAMA Nº 3

Tenemos una tabla en memoria de 10 frecuencias, la posición 1 corresponde al nº 0, la 2 al 1, etc.,
Por la lectora de fichas entra una serie de ellas con un nº del 0 al 9. Hacer un organigrama para obtener en las posiciones correspondientes de la tabla el nº correspondiente de cifras de él leidas.

### ORGANIGRAMA Nº 4

Tenemos 2 tablas en memoria una llamada VENTAS y otra NOMBRE, la primera contienen el total de ventas de una serie de vendedores, cada posición de la tabla corresponde a un vendedor y hay 20. La 2ª contiene el NOMBRE de cada vendedor en el mismo orden que la 1ª tabla.
Obtener el organigrama correspondiente para listar el NOMBRE de aquellos vendedores que hayan superado las 100.000 ptas. de ventas y su posición en la tabla.

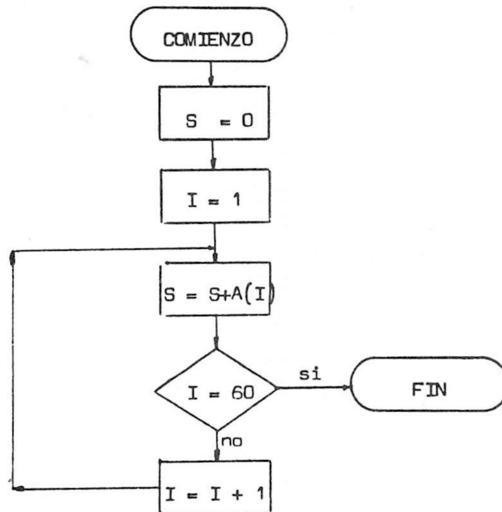
## PROBLEMA 4

Dados 60 números en memoria, se desea sumarlos y almacenar la suma en S.

## RESOLUCION

Un proceso lineal no seria útil en este caso, pues nos obligaria a detallar explicitamente las 60 sumas o al menos los 60 sumandos.Más adecuado vemos en este caso un proceso cíclico :

Suponemos que los 60 números forman un conjunto $A_i$.

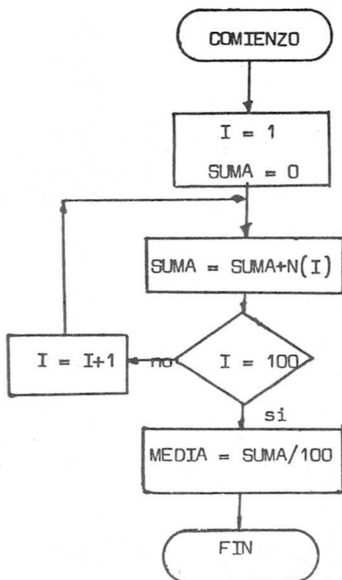



Vemos que el proceso se reduce a crear un ciclo en el cada pasada se incrementa un nuevo elemento del conjunto $A_i$ a la suma S, y que terminará cuando el subíndice valga 60, esto es : cuando hayamos sumado el último elemento.

## PROBLEMA 5

Dado un conjunto N de 100 elementos N(1), N(2), N(3)....... N(100), que suponemos en memoria, se pide, un organigrama que describa el proceso necesario para almacenar en SUMA la suma de los 100 elementos y en MEDIA la media aritmética de ellos.

## RESOLUCION

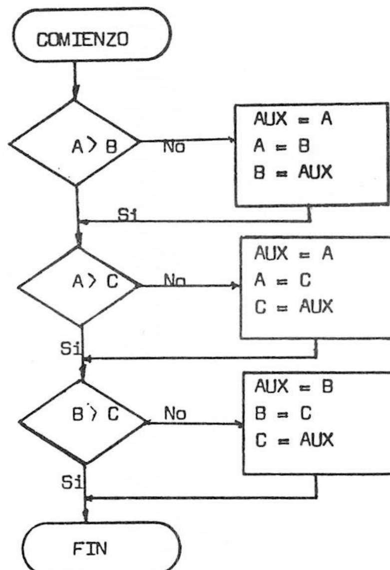


## PROBLEMA 6

Se tienen tres campos A, B, y C en memoria. Se desea un organigrama para compararlos entre si, e intercambiarlos de forma que el mayor se situe en A, el intermedio en B y el menor en C.

## RESOLUCION

El proceso consistirá en comparar cada número con los otros dos. Si en cada comparación están correctamente situados, se dejan como están, si no se cambian entre si. Obsérvese que para intercambiar el valor de dos campos es necesario usar un campo auxiliar.

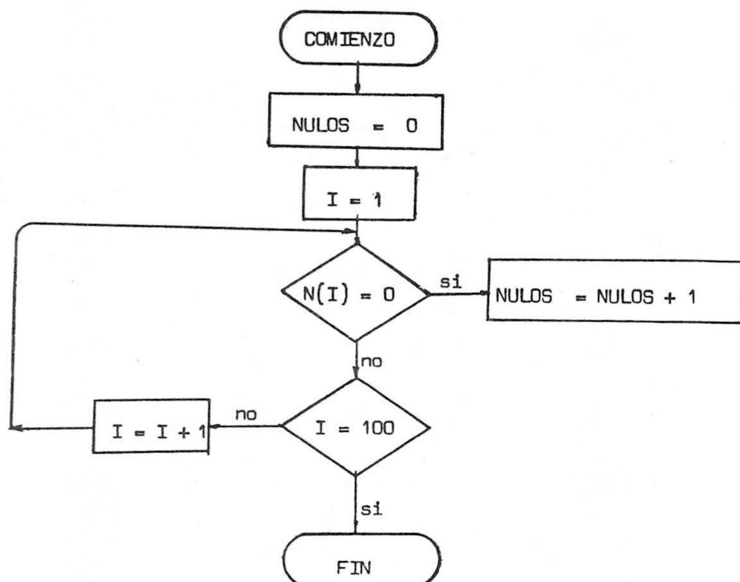


## PROBLEMA 7

Dado un conjunto N de 100 elementos N(1), N(2)........... N(100) que suponemos en memoria, se pide un organigrama para obtener el número total de elementos nulos y almacenar este valor en NULOS.

## RESOLUCION

El procedimiento será comparar cada uno de los 100 elementos del conjunto con 0. Cada vez que encontramos un elemento 0, sumamos 1 a NULOS con lo cual al final tendremos en NULOS el número total de ceros.



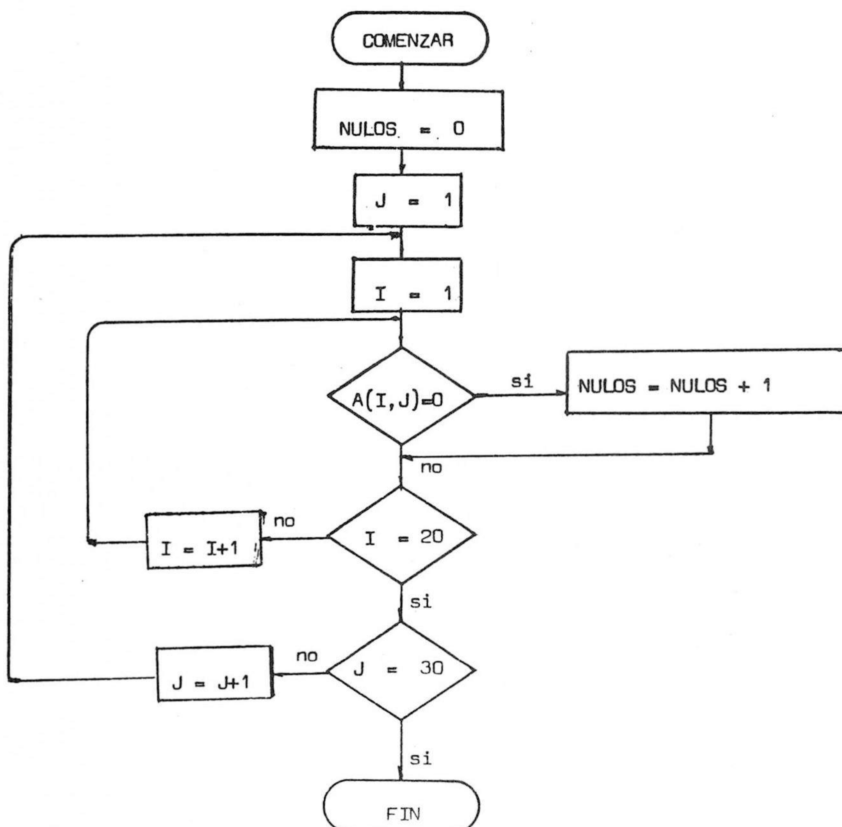

El problema siguiente es una aplicación a conjuntos de dos dimensiones del mismo problema.

## PROBLEMA 8

Dado un conjunto de dos dimensiones, definido por M(20,30), que suponemos en memoria, se pide un organigrama para obtener el número total de elementos nulos del conjunto, y almacenar este valor en NULOS.

RESOLUCION

El proceso será análogo al anterior.



## PROBLEMA 9

Dada una matriz de dimensión (20,30), se desea un organigrama para transponerla, es decir para cambiar las "filas"por las "columnas".

## ORGANIGRAMA Nº 1

Cierta empresa perfora en fichas sus operaciones de venta indicando el número del vendedor y el importe de la venta. La empresa tiene 100 vendedores numeradps de 1-100. Sequiere calcular e imprimir el total de ventas de cada vendedor.
Analizar el problema y hacer un ordinograma para resolverlo.

## ORGANIGRAMA Nº 2

Una empresa perfota sus ventas en fichas a razón de diez valores en cada ficha.
La última ficha de datos del porque puede no estar completa. En este caso el último campo valido irá seguido de un campo a nueves.
Se quiere calcular e imprimir el tottal de las ventas.
Analizar el problema y hacer un ordinograma para resolverlo.

## ORGANIGRAMA Nº 3

Ordenar una tabla unidimensional simple, TABLA de cien elementos, en secuencia creciente.
Hacer un ordinograma para conseguirlo.

## ORGANIGRAMA Nº 4

Se dispone en memoria de un ordenador de una tabla multiple con dos entradas: nº de articulo y precio unitario.
La empresa perfora en fichas sus ventas, a razón de una ficha por venta. En las fichas consta el nº de cliente, y la cantidad en unidades de venta, número de artículo. Las fichas vienen clasificadas y ordenadas por el nº de cliente. Calcular e imprimir los totales de ventas a cada cliente, imprimiendocada factura en pagina nueva, - controlando el listado.

La tabla ni está ordenada ni es coincidente, tiene 100 elementos.
Si alguna ficha lleva un artículo que no está en la tabla, mandar un mensaje de error y acabar.
Analizar el problema, y hacer un ordinograma para resolverlo.

ORGANIGRAMA Nº 5

Un banco tiene en la memoria de su ordenador una tabla multiple con deos entradas: nº de cuenta corriente y saldo de sas cuenta.
Se procesa un archivo en fichas con las operaciones de ingresos o pagos. Las fichas tieeen 3 campos: nº de cuenta, importe, código de la operación. Este código será una I para ingresos y una P para pagos.
Las fichas entrasn clasificadas y ordenadas según el nº de cuenta.
La tabla está ordenada por nº de cuenta, en consecuencia creciente, y tiene 5000 elementos.
Se quiere actualizar la tabla e imprimirla controlando el listado.
Analizar el problema y hacer un ordinograma para resolverlo.

ORGANIGRAMA Nº 6

Ordenar en secuencia decreciente, por número de artículo, una tabla multiple de 3 entradas: nº del cliente, nº de artículo, y departamento.
La tabla tiene 100 elementos.
Imprimirla ordenada.
Realizar un ordinograma para ello.

## PROBLEMA 10

Dada una matriz A en memoria, de dimensión A(20.30) y otra B, tambien en memoria, de dimensión B(20,30), se trata de realizar un organigrama para obtener otra matriz C tambien en memoria, de dimensión C(20,30) que sea la matriz suma de la A y de B.

## PROBLEMA 11

Dado un conjunto A en memoria de dos dimensiones A(40,50) y otro conjunto B, tambien en memoria , de una sola dimensión B(2000), se trata de obtener un organigrama que describa el proceso necesario para almacenar los elementos del conjunto A en el B.

## PROBLEMA 12

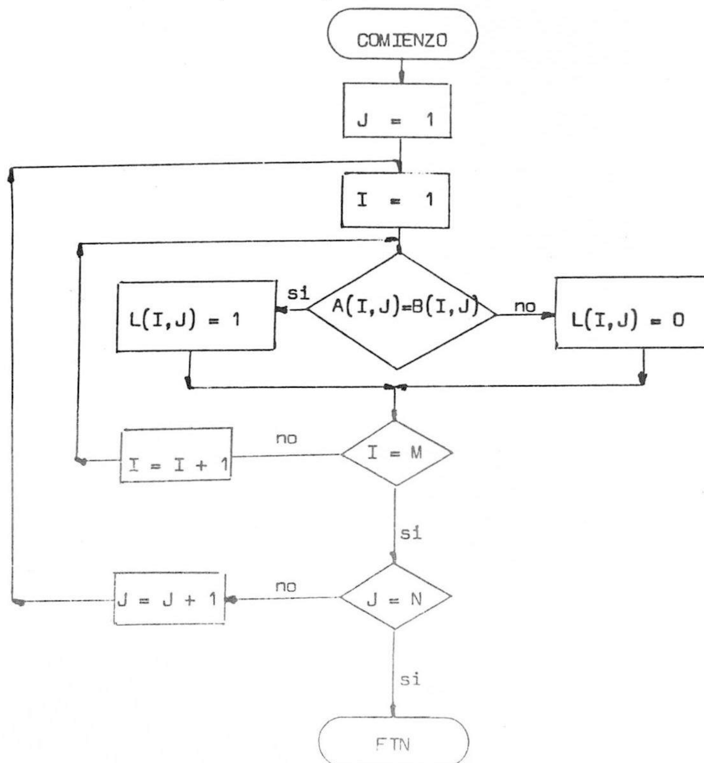
Dados dos conjuntos de dos dimensiones A(M,N) y B(M,N) hacer un organigrama para obtener un tercer conjunto L(M,N) de la forma siguiente :

Si $a_{ij} = b_{ij}$ entonces $l_{ij} = 1$

Si $a_{ij} \neq b_{ij}$ entonces $l_{ij} = 0$

RESOLUCION

Vemos que se trata de obtener la matriz logica, resultado de la comparación de las matrices A y B.

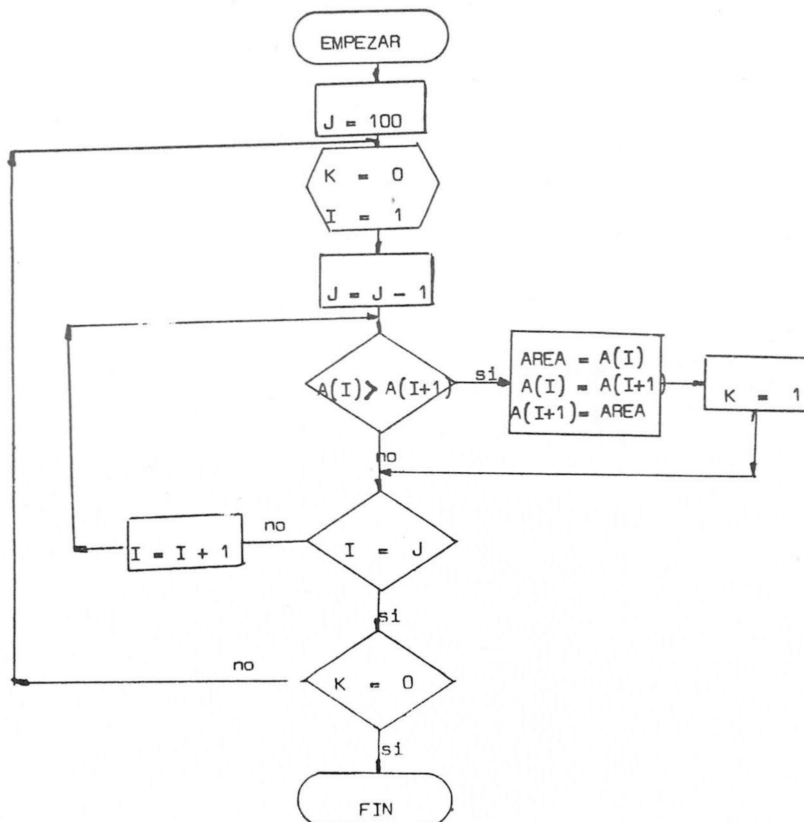


## PROBLEMA 13

Dado un conjunto A de 100 elementos , ya en memoria, hacer el organigrama del proceso necesario para ordenar los elementos de este conjunto.

### RESOLUCION

El procedimiento a seguir es el de comparaciones sucesivas. Si no se conoce puede verse la explicación que sigue a este organigrama.



EXPLICACION

Existen muchos procedimientos para ordenar los elementos de un conjunto . El procedimiento que vamos a seguir es el siguiente :

Vamos a ir comparando cada elemento con el siguiente de la tabla. Si al comparar una pareja de elementos ($A_i$ y $A_{i+1}$), estan bien ordenados,(en este caso $A_i < A_{i+1}$), se dejan en el orden que estan, pero si no es así,se cambian entre si.

Haciendo esto en toda la tabla(en una "pasada") conseguiremos llevar el mayor elemento al último lugar. Repitiendo la operación (en una nueva pasada) conseguiremos llevar el inmediatamente inferior al penúltimo y así sucesivamente conseguiremos ordenar la tabla.

¿Cual será el número de pasadas que debemos hacer?.
"A Priori" podriamos pensar que si N es el número de elementos de la tabla el número de pasadas deberá ser N-1. Sin duda que haciendo N-1 pasadas el conjunto quedará ordenado, pero es facil que no sean necesarias estas N-1 pasadas, y que el conjunto quede ya ordenado en lapasada K, siendo K < N-1, con lo cual las N-1-K últimas pasadas serian inútiles.

Debemos pues, de alguna forma detectar en que pasada ha quedado ordenado el conjunto. En este caso lo hacemos con el switch K, que al iniciar cada pasada lo ponemos a 0, y al efectuar un cambio lo ponemos a 1. Si al final de una pasada K mantiene el valor cero, es porque no se ha hecho ningún cambio, lo cual indicará que el conjunto está ya ordenado. Además podemos optimizar el proceso, haciendo en cada pasada una comparación menos, ya que los elementos finales van quedando ordenados y por lo tanto no es necesario compararlos.

Veamoslo mejor con un ejemplo :

Vamos a ordenar el siguiente conjunto formado por 10 elementos :

2, 6, 1, 10, 13, 6, 12, 7, 9, 4

1ª Pasada(9 comparaciones) : 2, 1, 6, 10, 6, 12, 7, 9, 4, 13
2ª Pasada(8 comparaciones) : 1, 2, 6, 6, 10, 7, 9, 4, 12, 13
3ª Pasada(7 comparaciones) : 1, 2, 6, 6, 7, 9, 4, 10, 12, 13
4ª Pasada(6 comparaciones) : 1, 2, 6, 6, 7, 4, 9, 10, 12, 13
5ª Pasada(5 comparaciones) : 1, 2, 6, 6, 4, 7, 9, 10, 12, 13
6ª Pasada(4 comparaciones) : 1, 2, 6, 4, 6, 7, 9, 10, 12, 13
7ª Pasada(3 comparaciones) : 1, 2, 4, 6, 6, 7, 9, 10, 12, 13

## PROBLEMA 14

Dadas 100 números almacenados en un conjunto en memoria N(1), N(2), N(3), N(4),.......... N(100), se desea un organigrama para sumarlos e imprimir la suma.

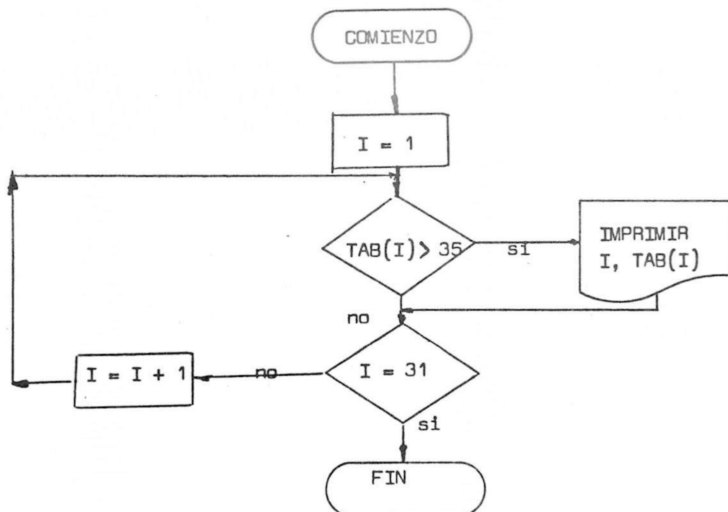
## PROBLEMA 15

Dado un conjunto en memoria de 100 elementos N(1), N(2), N(3)..... ....... N(100) se desea un organigrama para sumarlos e imprimir todas las sumas parciales y la total.

## PROBLEMA 16

Se tiene en memoria una tabla en la que figuran las temperaturas mínimas de Madrid en cada uno de los 31 dias del mes de Agosto. Se pide un organigrama para seleccionar aquellas temperaturas mayores de 35º e imprimirlas con los dias en que se produjerón.

### RESOLUCION



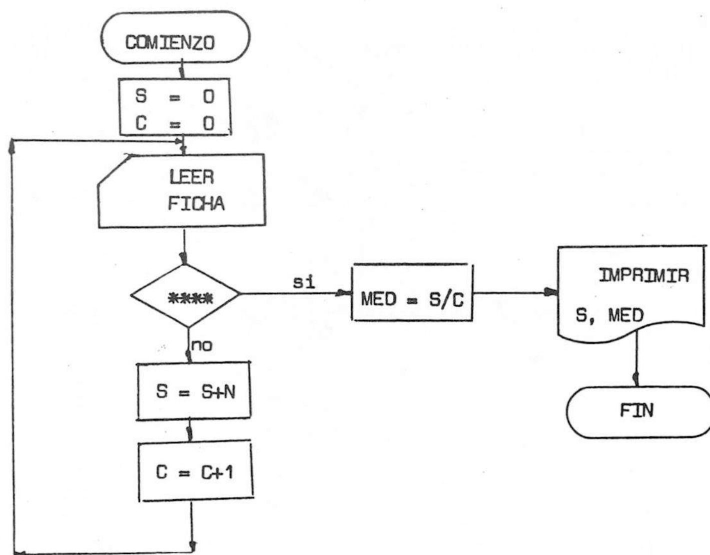
## PROBLEMA 17

Se tiene una serie de números perforados cada uno en una ficha. Se desea un organigrama para leerlos, calcular su suma, su media aritmética e imprimir ambas.

El fin del archivo se detectará por una tarjeta que llevará perforado **** .

### RESOLUCION

Lo normal es que no se sepa de antemano el número total de valores que van a introducirse, como ocurre en este caso. Pero detectaremos el final del archivo al leer la tarjeta ****. En este caso no habrá que contar el número de valores que se leen para calcular la media.




Normalmente no es necesario que dotemos al archivo de una tarjeta para detectar el fin, ya que la mayoria de los sistemas detectan automaticamente el fin del archivo con la presencia de una tarjeta de control especial para ello. En lo sucesivo no se indicará ninguna tarjeta para fin, ya que suponemos que el fin se detecta de forma standard.

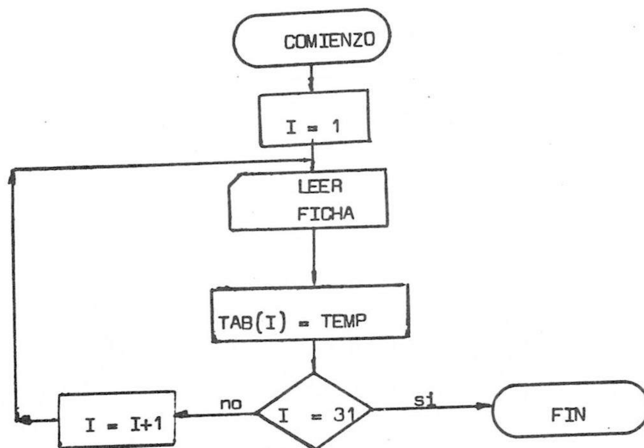
## PROBLEMA 18

Se trata de almacenar en una tabla en memoria, las temperaturas mínimas producidas en Madrid durante el mes de Agosto. Hay que almacenar en el primer elemento la temperatura del dia 1, en el segundo la del 2 y así sucesivamente, sabiendo que la temperatura de cada dia viene perforada en una ficha, estando estas fichas ordenadas por dia.

### RESOLUCION

Puesto que las fichas estan ordenadas de la misma forma en que han de almacenarse, la resolución es inmediata.

Si llamamos a la tabla de memoria TAB, y a la temperatura de cada ficha TEMP.




Si no hubieramos conocido el mes, es decir que si no se supiera de antemano el número de fichas que han de leerse, en vez de llegar al FIN cuando I= 31, hubieramos preguntado después de la orden de lectura se es el fin del archivo, y en caso afirmativo ejecutariamos el fin.

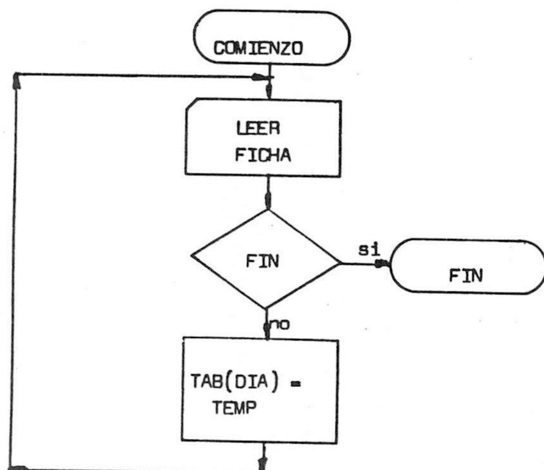
## PROBLEMA 19

Se trata de obtener en memoria la misma tabla del problema anterior pero ahora las fichas no estan ordenadas por dia, sino que no guardan ningún orden. Sin embargo ahora cada ficha lleva perforado además de la temperatura el dia en que se ha producido.

### RESOLUCION

Ahora al leer una ficha obtenemos como información la temperatura y el dia , es decir, el elemento de la tabla en el que ha de guardarse esa temperatura.

Si llamamos a la tabla de memoria TAB, y a los dos datos de cada ficha DIA y TEMP.




Observese que ahora hemos detectado el fin de forma automatica, imdependientemente del número de fichas que haya.

## PROBLEMA 20

Hacer un organigrama para multiplicar dos números A y B teniendo en cuenta que el ordenador disponible solo suma y resta. Los números vienen en una tarjeta.

## PROBLEMA 21

Se perforan en fichas las ventas por año de una cierta empresa. Su supone que estas ventas corresponden a un estudio de mercado y que son previsiones,
REalizar un organigrama para obtener el total de ventas para 10 años, teniendo en cuenta que simel año es impar al las ventas totales de ése año le descontaremos un 2% y si el año es bisiesto le aumentaremos un 5%.

## PROBLEMA 22

Un colegio perfora en fichas las caracteristicas de sus alumnos, tales como: nº de matricula, edad, curso, dirección, etc. Se quiere contar el número de alumnos que tiene el cologio y obtener impreso dicho número. Analizar el problema y hacer un ordinagrama para resolverlo.

## PROBLEMA 23

Un fichero en fichas contiene en cada una de ellas, número de empleados y horas trabajadas por él. Se quiere listar las fichas poniendo un + junto aquellas que tengan más de 40 horas trabajadas, imprimiendo además al final el número total de los que tengan +. Analizar el problema y hacer un orginagrama para resolverlo.

## PROBLEMA 24

Un fichero de 130 fichas, contiene dos tipos de fichas: unas lle van perforado el nombre el empleado, y otras la dirección del mis mo. Cada ficha de nombre va seguida de su correspondiente ficha de dirección. Listar las 40 fichas primeras, colocando el nombre y la dirección de cada empleado en una linea. Se supone que la ordenación de las fichas escorrecta. Analizar el problema y hacer un or dinagrama para resolverlo.

## PROBLEMA 25

Una empresa perfora en fichas el importe de sus ventas. Cada ficha contiene el importe de una sola venta. Se quiere calcular e impri mir el total vendido. Analizar el problema y hacer un ordinagrama para resolverlo.

## PROBLEMA 26

Un colegio perfora en fichas los datos de sus alumnos. Entre estos datos esta la altura de sus alumnos. SE quiere calcular la altura media del colegio e imprimirlo.

## PROBLEMA 27

Se quiere obtener un listado del año y de la cantidad anual acumulada por un capital colocado en un banco al 7% de interés compuesto anual, durante los 25 años siguientes al año de ingreso. Con con trol de listado. Para ello se dispone de una ficha en la que consta el capital inicial y el año de ingreso en el banco. Se desea contro lar el listado. Analizar el problema y realizar un ordinograma para resolverlo.

## PROBLEMA 28

Se tiene un archivo en fichas. En cada una esta perforado, además de otras cosas, un 1, un 2, ó un 3, según la persona a quien corresponda esa ficha sea:

1 ----------- casado
2 ----------- soltero
3 ----------- otro estado

Se trata de imprimir el número total de personas de cada unp de los tres estados. Analizar el problema y hacer un ordinograma para resolverlo.

## PROBLEMA 29

Se quieren obtener las órdenes de pedido en un almacén a partir de un fichero de existencias en cinta. Los registros de la cinta tienen cada uno los 5 campos sigüientes:

- Nº de articulo. N A
- Cantidad pedida en el antepenúltimo mes. $C_1$
- Cantidad pedida en el penúltimo mes. $C_2$
- Cantidad pedida en el último mes. $C_3$
- Cantidad existente en almacén. Stock.

Existe un registro por cada articulo existente en almacén.
Se trata de hallar la cantidad supuesta de pedido para el próximo mes, que será la media de las pedidas en los tres últimos meses, y compararla con la existente en almacén. Si es mayor, <u>se imprime una línea, indicando el nº de artículo, y la cantidad a pedir</u>, que será:

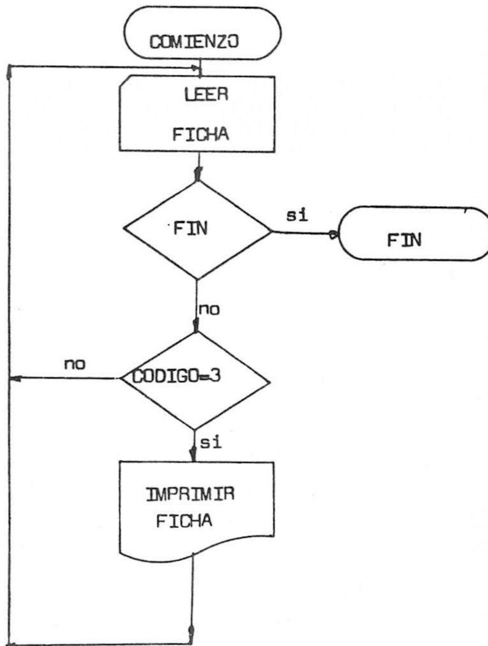
C. Pedida = 1,3 (Cantidad supuesta - Existencias)

Analizar el problema y hacer un ordinograma para resolverlo.

## PROBLEMA 30

Dado un fichero en tarjetas imprimir todas aquellas tarjetas que tengan como código un 3.

RESOLUCION

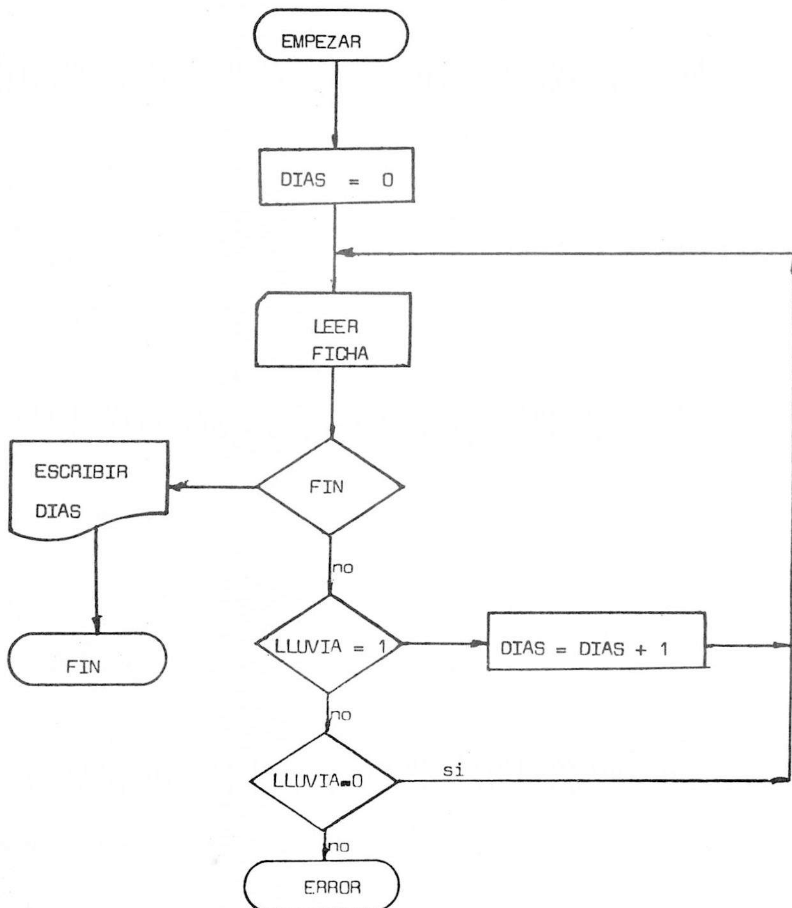


## PROBLEMA 31

En un observatorio meteorologico se perfora cada dia una ficha en la que entre otros datos figura un campo llamado LLUVIA que vale 1 si llovió y 0 si no llovió ese dia.

Hacer un organigrama que describa el proceso para calcular el nº de dias que llovió durante el mes, y que imprima este número.

RESOLUCION

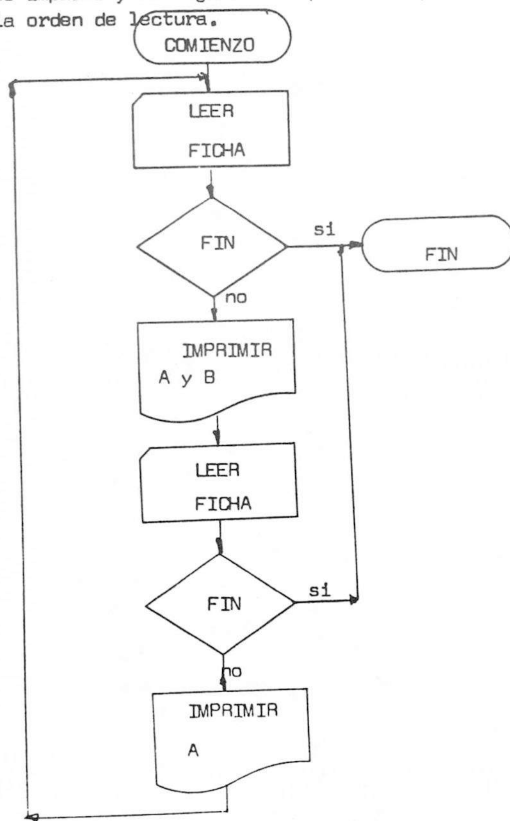


# PROBLEMA 32

Se tiene un paquete de fichas en las que hay perforado dos campos A y B. Hacer un organigrama para listar el campo A de las fichas pares y los dos campos A y B de las impares.

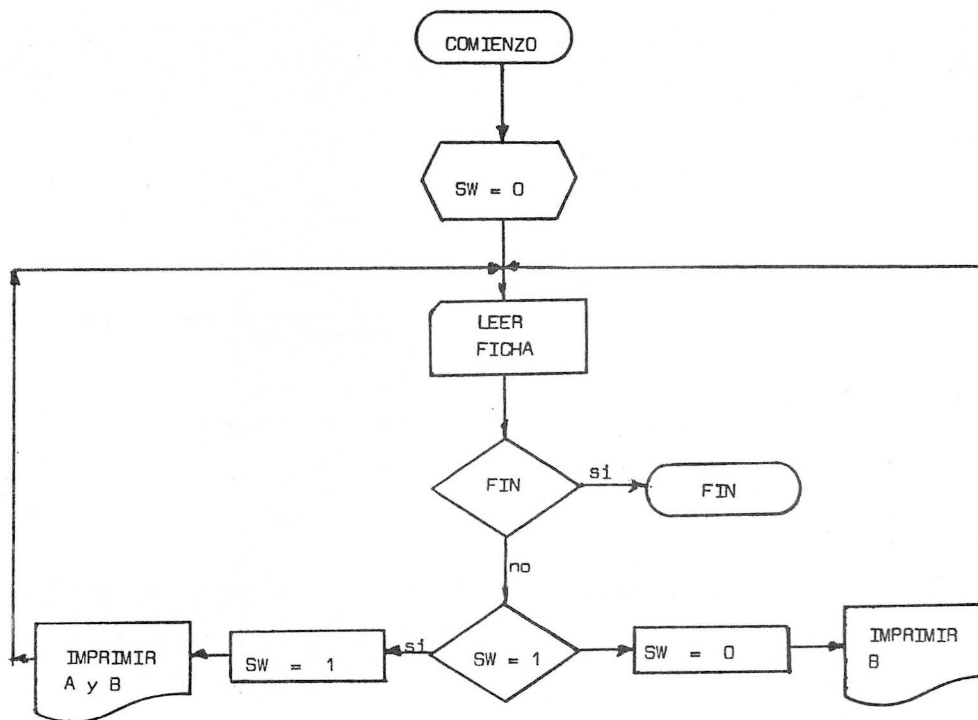
## RESOLUCION

La primera solución consiste en útilizar dos ordenes de lectura en puntos distintos, y encerrarlas en un ciclo. De esta forma la primera lectura leerá las fichas impares y la segunda las pares. Despues veremos otra solución con una sola orden de lectura.




La segunda solución solo se utiliza una orden de lectura, dentro de un ciclo.

Entonces ahora de alguna forma tendremos que distinguir cada vez que se ejecute siha sido leida una ficha par o impar. Observese bien de que forma se hace.




Ahora en vez de describir dos ordenes de lectura solo describimos una y el problema lo resolveremos mediante la variable SW, a la que solo le permitimos que tome 2 valores (el 0 y el 1) y que nos permite de esta forma elegir un camino entre dos, según el valor que tenga, valor que nosotros controlaremos adecuadamente.

A una variable que actua de esta forma en un proceso se le suele llamar SWITCH.

## PROBLEMA 33

En una empresa se perfora una ficha por cada empleado, en la que entre otros datos figura el año de nacimiento. Hacer un organigrama para contar el número de empleados que nacieron en el año 1945, e imprimir este.

## PROBLEMA 34

Se tiene un stock archivado en cinta magnética. Cada registro tiene: nº de piezas, stock de ella y tres campos más en los que están los consumos de esas piezas en los tres últimos meses.
Se trata de preparar un archivo de pedidos en fichas según las reglas:

1ª. Si el promedio del consumo de una pieza en los tres últimos meses es mayor que el stock, perforar una ficha en la que se indique:

- El Promedio.
- El Stock
- La diferencia entre ambos.
- El número de la pieza.

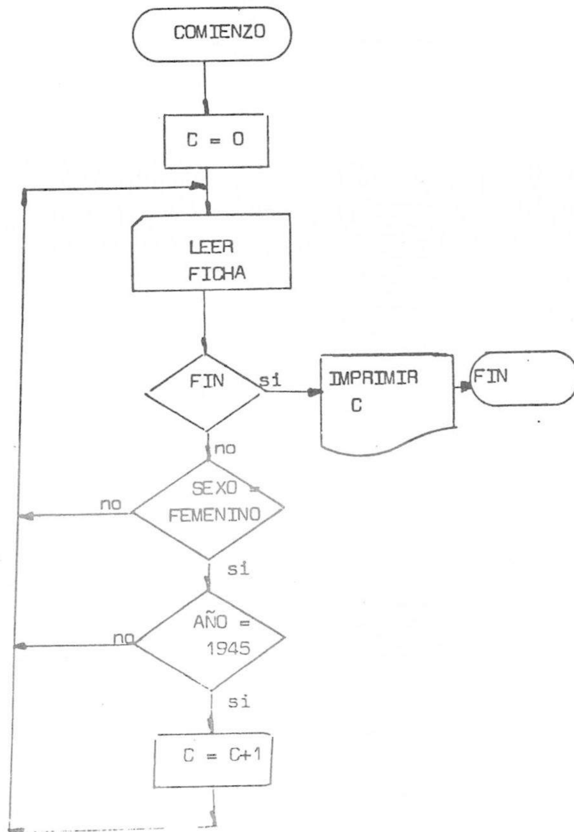
2ª. Los registros que cumplan la condición anterior deberan listarse por impresora sin control de listado y los que no la cumplen se ignoran.
Analizar el problema y hacer un ordinograma para resolverlo.

## PROBLEMA 35

En una empresa se perfora una ficha por cada empleado en la que entre otros datos figura la fecha de nacimiento y el sexo. Hacer el organigrama para contar el número de empleados de sexo femenino nacidos en el año 1945.

### RESOLUCION

Solo contaremos aquellos empleados que reunan las dos condiciones.



## PROBLEMA 36

En una empresa se perfora una ficha para cada empleado en la que entre otros datos figura el sexo y el sueldo de cada empleado, se desea un organigrama para contar en número de empleados varones que perciben más de 30.000 pts. al mes.

## PROBLEMA 37

Una empresa tiene un fichero en cinta, secuencial por número de empleado, de todos sus empleados. En cada registro hay, entre otros campos, uno con el número de empleado, otro con el nombre y un codigo:

1 ------------ empleado casado.
2.------------ empleado soltero
3 ------------ empleado otros estados.

Se quiere listar en la impresora, secuencialmente, el número y el nombre de cada empleado, indicando también si es casado, soltero u otro estado, teniendo en cuenta:

a) Hay que poner cabeceras y controlar el salto de página.
b) Si un registro del fichero esta fuera de secuencia, se da un mensaje por la consola indicandolo, y se ignora el registro.
c) Si el Código de un registro no es uno de los indicados, se da otro menseje por consola y se ignora también el registro, aunque se tiene en cuenta su número de empleado para la secuencia.

Analizar el problema y hacer on ordinograma para resolverlo.

## PROBLEMA 38

Se tiene un lote de fichas ordenado según el campo del número de matricula, en secuencia creciente. Los números de matrícula no son consecutivos.

Se quiere comprobar la secuencia y si alguna ficha esta desordenada imprimir su contenido en un listado de errores.

Analizar el problema y hacer un ordinograma para resolverlo.

## PROBLEMA 39

En una empresa se perfora una ficha por empleado en la que entre otros datos figura el nombre y el sueldo mensual. Se pide un organigrama para obtener una relación de aquellos empleados que perciben más de 30.000 pts, al mes.

## PROBLEMA 40

Se tienen en fichas las operaciones ingresos o pagos realizadas por los clientes de una sucursal bancaria. El formato de las fichas es:

| | | | |
|---|---|---|---|
| Nº Cliente | Importe | Operación | . . . . . |

El importe siempre es positivo, y la operación es realizar ingreso o deducción.
Se quiere obtener una cinta magnetica en la que figuren por cada cliente el número del mismo y el saldo de sus operaciones.
Las fichas estan agrupadas por números de cliente. Los números de cliente van de 1-100, y hay como mínimo una ficha por cliente.
Analizar el problema y hacer un ordinagrama para resolverlo.

## PROBLEMA 41

En una empresa se perfora una ficha para cada empleado en la que entre otros datos figura el sexo y el estado civil. Se desea un organi grama para contar :

- El número de varones.
- El número de mujeres casadas.

## PROBLEMA 42

A fin de ños hay que calcular e imprimir sin control de listado el total de ventas de cada mes, el promedio de ventas por mes y total anual de ventas.
Para ello se dispone de un fichero en el cual cada ficha corresponde a una venta y tiene perforado el mes en que se realizó la venta y el importe de la misma.
Las fichas se han clasificado por meses y se han ordenado en secuencia creciente de número de mes.
Puede haber meses que no tengan ventas. A todos los efectos estos meses no se consideraran.
Analizar el problema y hacer un ordinagrama para resolverlo.

## PROBLEMA 43

Un empresa perfora en fichas sus operaciones de venta.
Cada ficha tiene tres campos, para nº de cliente, cantidad en unidades del artículo vendido y precio unitario de dicho artículo.
Las fichas vienen clasificadas y ordenadas en secuencia creciente de nº de cliente. Cada cliente puede tener varias fichas.
Se trata de hacer una factura a cada cliente, en la que conste número de cliente e importe total a pagar.
Cada factura se imprimirá en página nueva con cabecera.
Analizar el problema y hacer un ordinagrama para resolverlo.

## PROBLEMA 44

Un colegio perfora en fichas la altura de sus alumnos existiendo en cada ficha , el nombre,el sexo, la altura de cada alumno. Se pide un organigrama para obtener una relación de aquellos alumnos varones que midan 2 metros o más con el fin de formar un equipo de baloncesto, debiendose imprimir tambien el número total de seleccionados.

## PROBLEMA 45

Se tiene en fichas las distancias de la Tierra a su satelite. En distintos puntos de su orbita. Se quiere hallar la distancia mínima e imprimirla.
Analizar el problema y hacer un ordinograma para resolverlo.

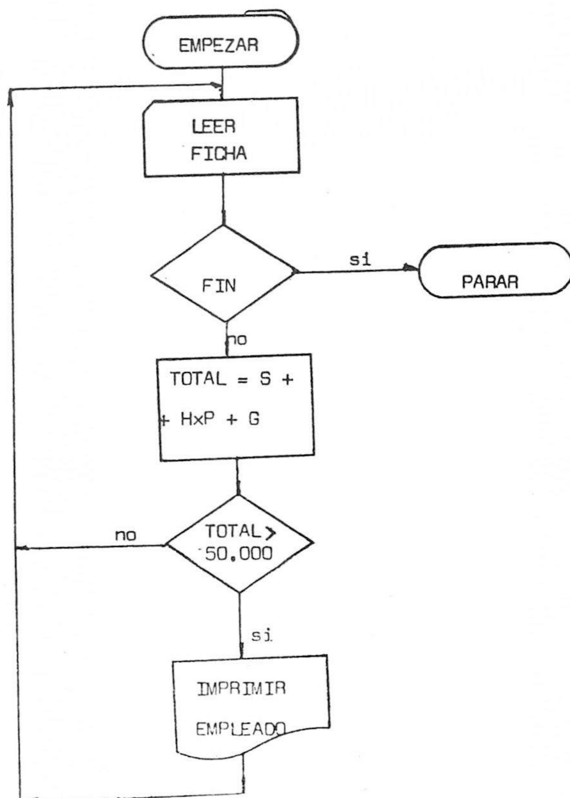
## PROBLEMA 46

En una empresa se perfora una ficha para cada empleado, en la entre otros datos, vienen :

El sueldo S
El número de horas extras H
El precio de cada hora extra P
Otros ingresos I

Hay que describir mediante un organigrama el proceso necesario para obtener a partir de estas tarjetas, una relación de todos aquellos empleados que por uno u otro concepto obtengan en total más de 50.000 pts.

RESOLUCION

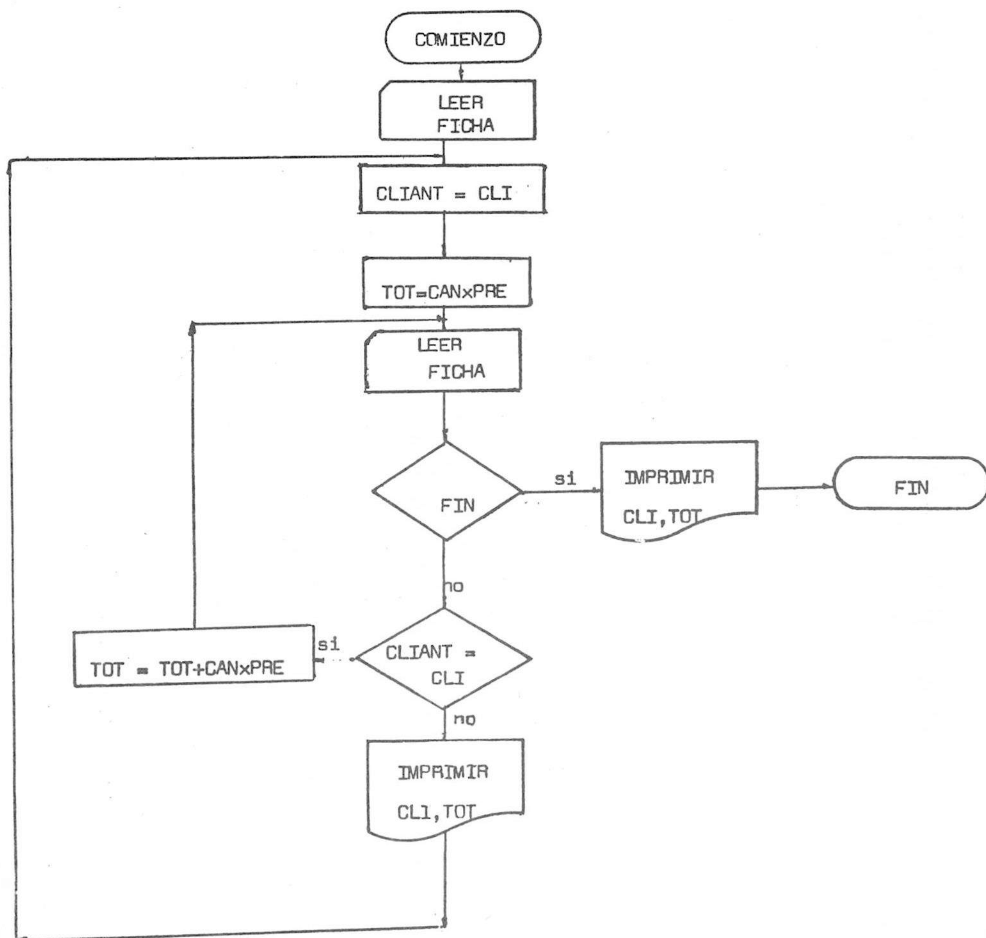


## PROBLEMA 47

Una empresa perfora una ficha por cada venta, en la que figura el nº de cliente, la cantidad de artículos y el precio unitario, entre otros datos. Estas fichas se clasifican por nº de cliente.

Se pide una relación de todos los clientes con el importe total a pagar por cada uno de ellos.

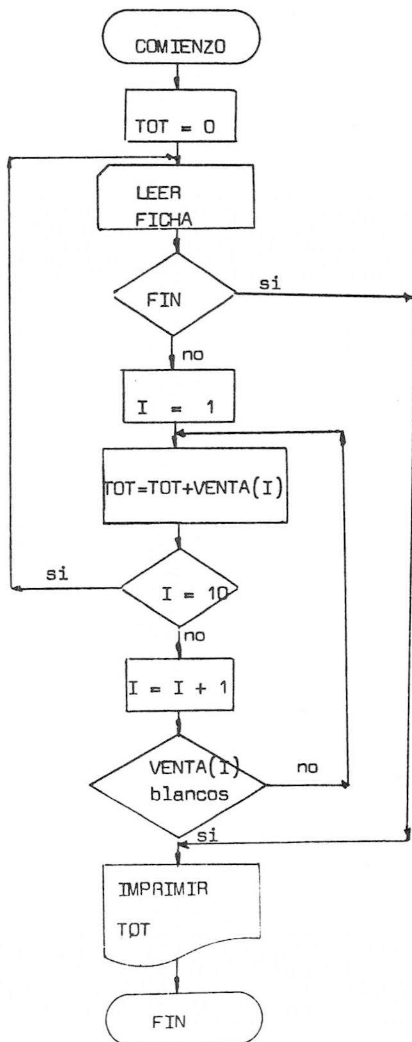
### RESOLUCION



## PROBLEMA 48

Una empresa perfura en fichas sus ventas, a razón de diez ventas en cada ficha. La última ficha puede no estar completa. Calcular e imprimir el total de ventas.

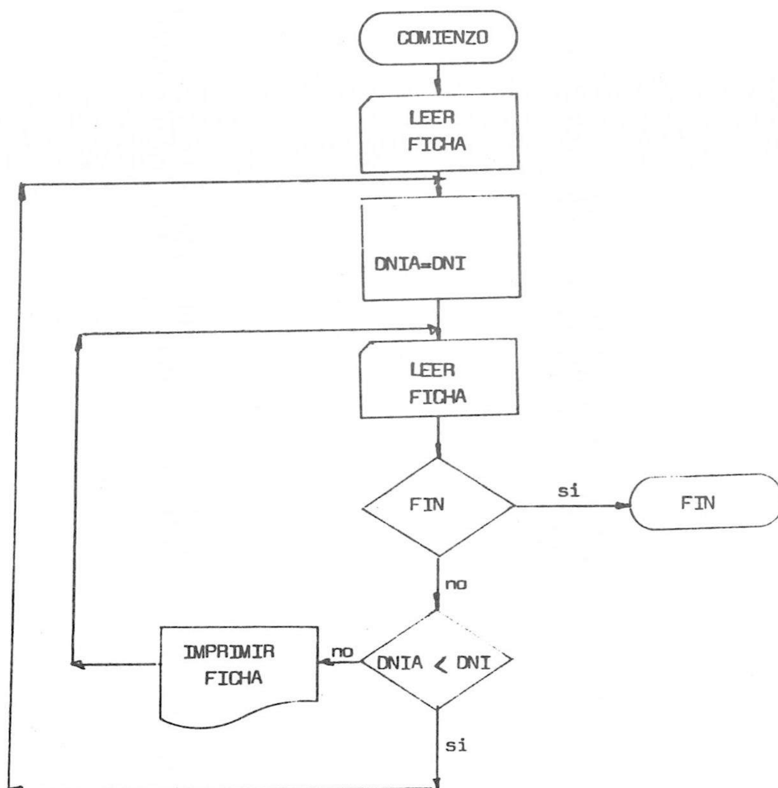
RESOLUCION



## PROBLEMA 49

En una empresa se tiene un archivo de empleados ordenado por D.N.I en orden creciente. Hacer un organigrama para comprobar la ordenación de este archivo. Si se detecta un registro desordenado, se imprimirá este.

RESOLUCION

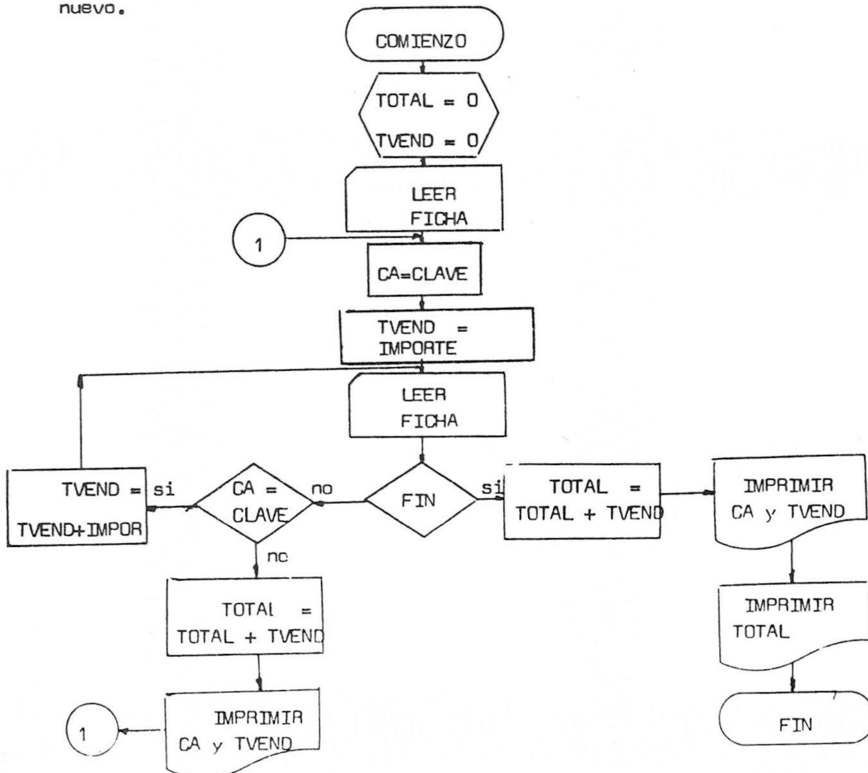


# PROBLEMA 50

En una empresa se perfora una ficha por cada venta que realiza cada vendedor, en la que figura la clave del vendedor y el importe. Con todas estas fichas, ordenadas por clave de vendedor, se obtiene un archivo de todas las ventas realizadas por la empresa en un determinado periodo de tiempo. Se pide un organigrama para obtener una relación de todas las ventas efectuadas por cada vendedor, asi como el total vendido por todos los vendedores.

## RESOLUCION

Como el archivo está ordenado por clave de vendedor, todas las ventas de un vendedor estarán seguidas. Para detectar el cambio de vendedor tendremos almacenada la clave del vendedor que se está procesando en un campo que llamremos CA, cuando se lea un vendedor con clave distinta de esta, habrán acabado las ventas del vendedor anterior y empezarán las del nuevo.

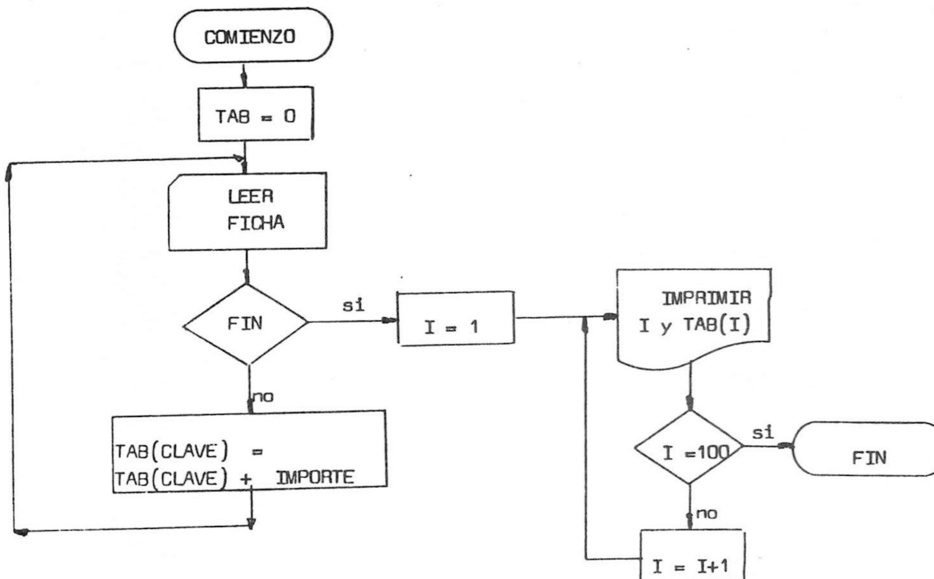


# PROBLEMA 51

Cierta empresa perfora en fichas las ventas realizadas por cada vendedor, en las que figura la clave del vendedor y el importe vendido. La empresa dispone de 100 vendedores, cuyas claves van del 1 al 100. Con todas las fichas sin ordenarlas se crea un archivo de ventas de la empresa. Se pide un organigrama para obtener una relación del total vendido por cada vendedor, asi como el total vendido por todos los vendedores.

## RESOLUCION

Observese ahora que al no estar ordenadas las fichas no podemos seguir el método anterior. Ahora forzosamente deberemos tener tantos totales como vendedores haya, esto es 100, y al leer cada ficha sumar el importe al total del vendedor correspondiente. Los totales los tendremos en una tabla que llamaremos TAB, formada por 100 elementos,



## PROBLEMA 52

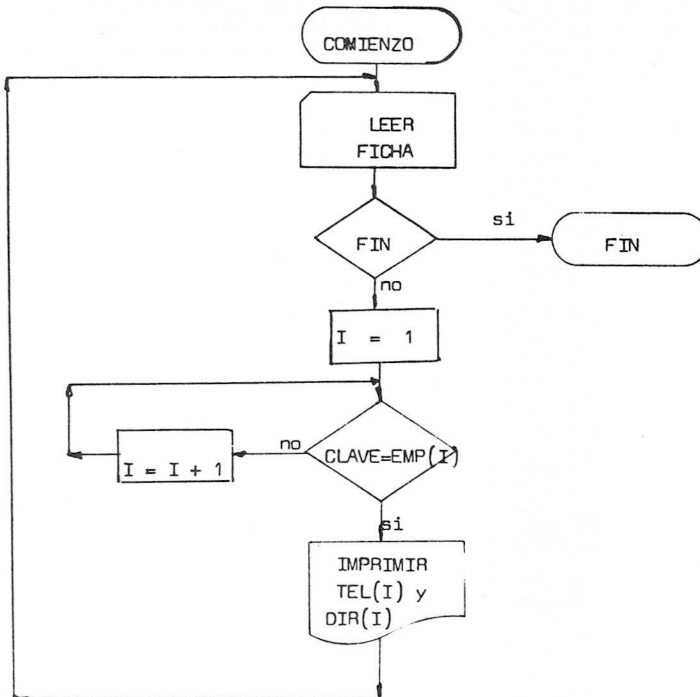
Se tiene en memoria tres tablas del mismo número de elementos cada una . La primera guarda claves de empleados, la segunda direcciones y la tercera teléfonos, de forma que los elementos de las tres tablas que ocupan el mismo lugar en cada tabla corresponden a la misma persona.

Por otro lado tenemos fichas en la que existe en cada una, una clave de empleado.

Se pide u n organigrama para listar la dirección y el teléfono de los empleados cuyas claves figuran en las fichas.

### RESOLUCION

Llamaremos a las tres tablas EMP, DIR y TEL respectivamente.

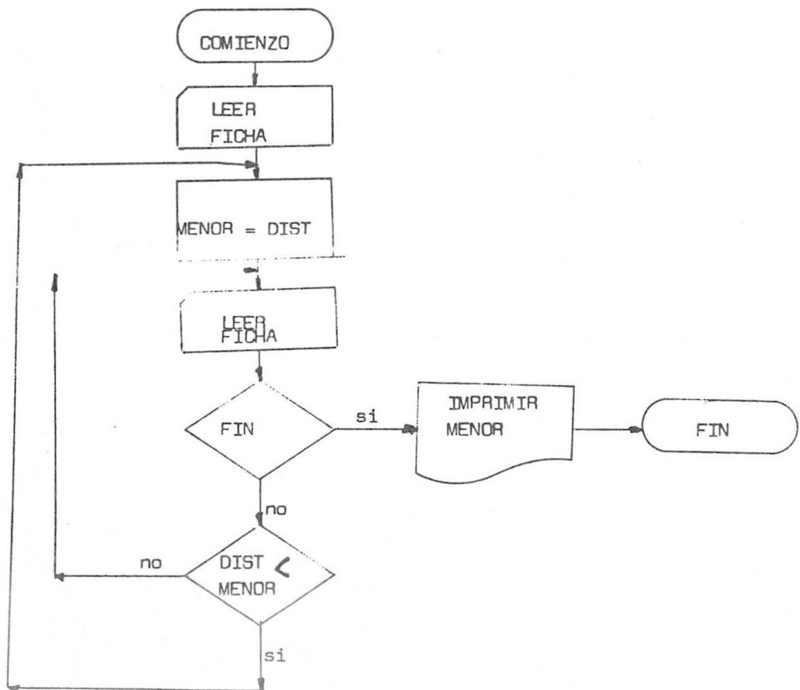


## PROBLEMA 53

Se tiene fichas en las que se halla perforada la distancia a la tierra de un satélite en distintos puntos de su orbíta. Se pide un organigrama para leer las fichas, calcular la mínima distancia, e imprimir esta.

### RESOLUCION

El proceso es inmediato. Cada vez que leamos una distancia menor a la mínima encontrada, hasta ese momento, la tomaremos como la menor. Inicialmente la menor será la primera leida.



## PROBLEMA 54

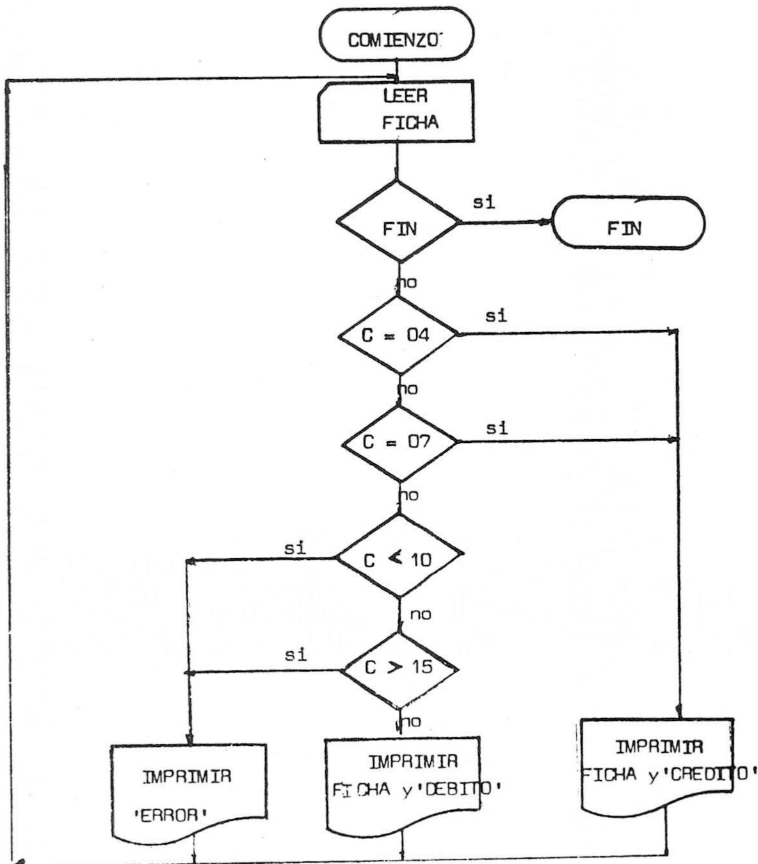
Existe un fichero en tarjetas en cada una de cuales debe existir en las dos primeras columnas uno de los códigos siguientes :

04, 07, 10, 11, 12, 13, 14, 15.

Se pide un organigrama para listar integramente las 80 posiciones de cada ficha, acompañadas de la palabra :

CREDITO : Si el código es de 04, ó 07.
DEBITO : Si el código es 10, 11, 12, 13, 14,ó15.
ERROR : En cualquier otro caso.

RESOLUCION



## PROBLEMA 55

En una empresa hay un fichero de empleados en tarjeta perforada en el que entre otros datos figura el sueldo y el nombre del empleado. Se ordena este archivo por sueldo en orden descendente y por nombre en orden ascendente. Se pide un organigrama para comprobar la validez de la secuencia debiendose listar todos aquellos registros que aparezcan fuera de orden.

## PROBLEMA 56

Una revista tiene el archivo de sus suscriptores grabado en cintas magneticas. Cada registro contiene nº de suscriptor y año en que expira la suscripción.

Se quiere:

1º. Grabar en otra cinta los suscriptores que siguen siendo a partir de 1975.

2º Imprimir los registros bajas de tres en tres.

3º Contabiliar las bajas y los que quedan.

Analizar el problema y hacer un ordinograma para resolverlo.

## PROBLEMA 57

Se dispone en cinta magnetica de los ingresos efectuados en una sucursal bancaria por sus clientes. En cada registro consta cantidad ingresada, mes y años del ingreso.

Se quiere obtener un tabulado con control los totales ingresados en la sucursal por mes y por año, durante cinco años, a partir del primer registro de la cinta.

En la cinta hay registros correspondientes a más de 5 años, y no tienen por que ser consecutivos no constar todos los meses. Pero todos los registros correspondientes a un año a un año estan agrupados en la cinta.

Analizar el problema, hacer un ordinagrama para resolverlo y comprobarlo.

## PROBLEMA 58

La compañia de electricidad factura de acuerdo con la siguiente tarifa : Los primeros 100Kw a 3 pesetas y los siguientes a 2 pesetas, Cada cliente tiene dos fichas una con la lectura de mes actual y la segunda con la del mes anterior. Se pide una relación de todos los clientes, con el importe que debe abonar cada uno de ellos.

## PROBLEMA 59

Se dispone en cinta magnetica de un archivo con el número de coches que han pasado por un punto determinado de la red nacional de autopistas. Cada registro tiene los siguientes campos: ano, mes, semana en que pasaron los coches y números de coches que pasaron.

Se quiere obtener en un tabulado, sin controlar, los totales de coches, por semana, mes y año durante el primer año a partir del primer registro de la cinta, calculando ademas el promedio de coches por mes.

En la cinta hay registros correspondientes a más de un año y no tienen porque ser consecutivos, ni existir todas las semanas y meses.

Todos lso registros de un mismo año estarán agrupados; dentro de ellos lo estarán por semanas y meses.

Analizar el problema y hacer un ordinagrama para resolverlo.

## PROBLEMA 60

La revista "Azul y Blanco" tiene en cinta maganetica un archivo de suscriptotes. Cada registro contiene: nº de suscriptor, y fecha en que expira la suscripción. Los registros estan clasificados en secuencia creciente de nº de suscriptor.

Se perfora en fichas las renovaciones de suscripción. Cada ficha contiene nº de suscriptor y nueva fecha de expiración.

Las fichas se procesan en consecuencia creciente de nº de suscriptor.

El proceso consiste en obtener una nueva cinta con las fechas definitivas de suscripción y el nº de suscriptor.

Los registros que no tengan ficha de renovación se graban tal como estan.

Toda ficha tiene registro, pero no todo registro tiene ficha.

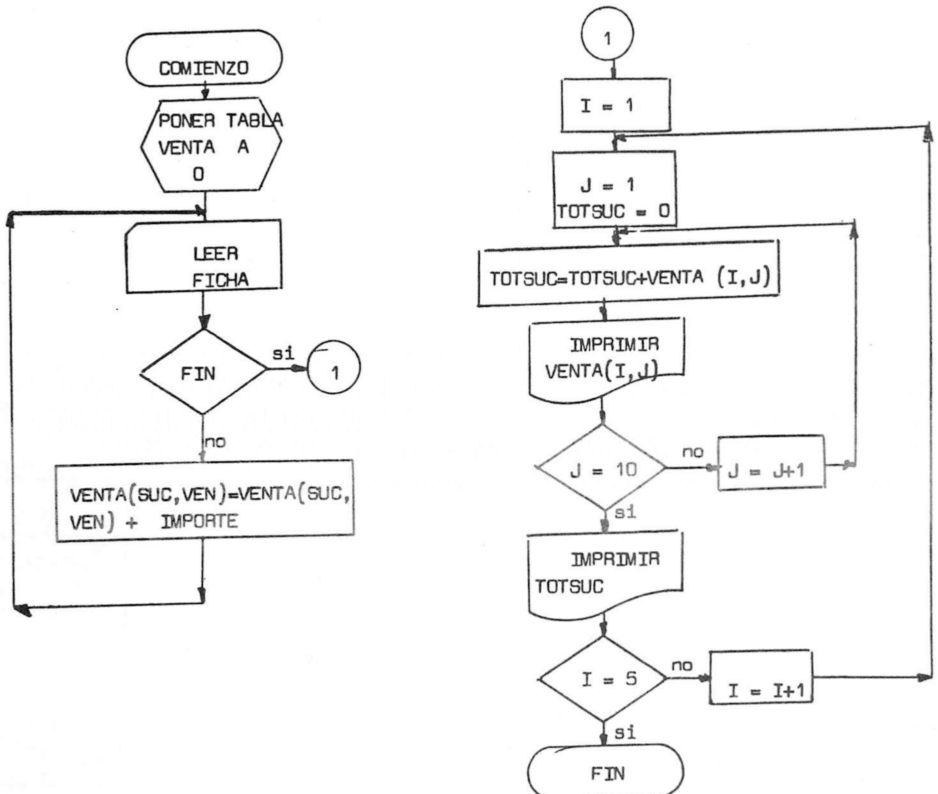
Analizar el problema y hacer un ordinograma para resolverlo.

## PROBLEMA 61

Una empresa tiene cinco sucursale númeradas del 1 al 5 y en cada una 10 vendedores númerados del 1 al 10. Sus ventas se perforan en fichas en las que figura el número de sucursal, nº de vendedor y el importe de la venta. Se pide un organigrama para a partir de estas fichas calcular e imprimir el total de ventas de cada vendedor y el total de cada sucursal, suponiendo que las fichas no se han clasificado.

### RESOLUCION

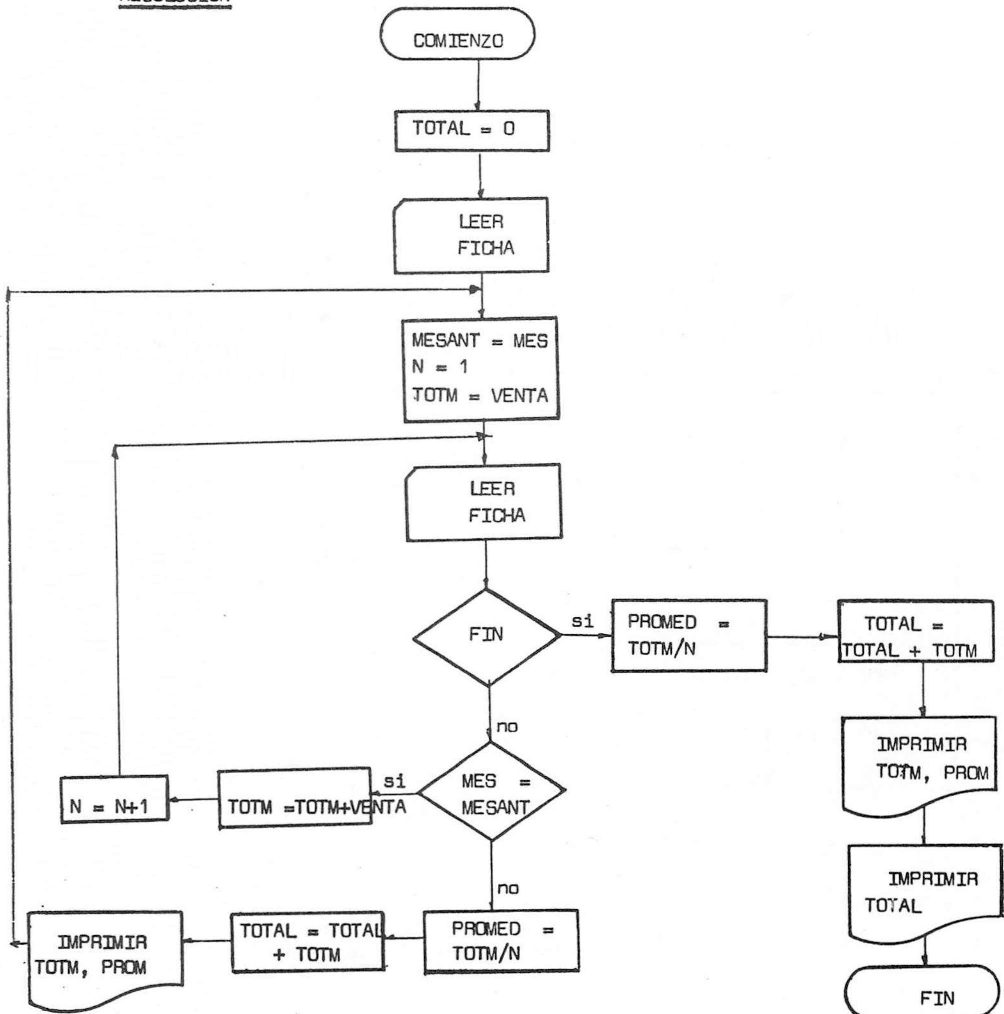
Con las ventas, de cada vendedor formaremos una tabla de 50 elementos (50 vendedores), de dos dimensiones VENTA(5, 10). El primer subíndice indicará el número de sucursal y el segundo el nº de vendedor, dentro de esa sucursal.



## PROBLEMA 62

A fin de año en una empresa hay que calcular e imprimir el total de ventas de cada mes, el promedio mensual de ventas y el total anual. Para cada venta hay perforada una ficha que indica el mes y el importe de la venta. Las fichas se han clasificado por meses en orden creciente. Hacer un organigrama para obtener los resultados pedidos.

RESOLUCION



## PROBLEMA 63



En una empresa existe un reloj automatico para la entrada, en el que al"fichar" cada empleado perfora una ficha con el número de empleado y la hora en que entra,(especificada en horas y minutos).

Con todas las fichas de un mes ordenadas por nº de empleado se crea un archivo , que permitirá descontar de la nómina de cada empleado lo correspondiente a los retrasos sufridos teniendo en cuenta que :

- La hora de entrada son las 8h,30m.
- Existe un margen de tolerancia de 15m.
- Si un individuo llega pasado este margen tiene un descuento de 40pts.
- Además, por cada minuto transcurrido de este margen se le descuentan 2 pts.

Hacer un organigrama para leer todas estas fichas, y obtener una relación de cada empleado con el importe en pts. que se le ha de descontar de la nómina, debido a su impuntualidad.

## RESOLUCION

El proceso a seguir podrá consistir en transformar la hora de llegada en minutos y restar la hora normal de entrada, tambien en minutos, que es : $8 \times 60 + 30 = 510$m.

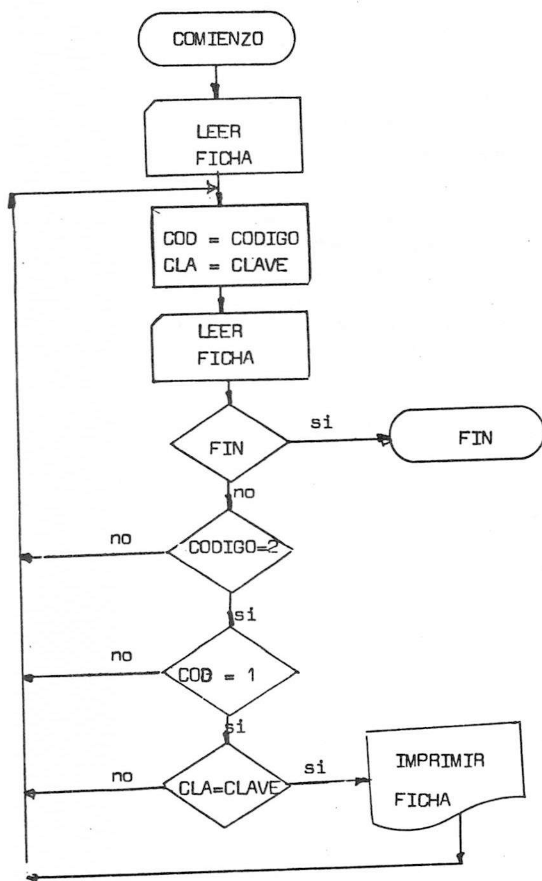
Si esta diferencia es menor o igual a 15, el empleado no tiene sanción. Si es mayor de 15 el empleado tiene 40 pts. de sanción más 2 pts. por cada minuto que ha transcurrido despues de estos 15.

Todo esto complementado con los correspondientes procesos de entrada y salida, de detección de cambio de empleado, etc.

## PROBLEMA 64

Tenemos un fichero en tarjetas, en cada una de las cuales hay una clave y un código. Se pide un organigrama para listar todas las fichas con código 2 que vayan precedidas por una ficha de código 1 de esa misma clave.

### RESOLUCION



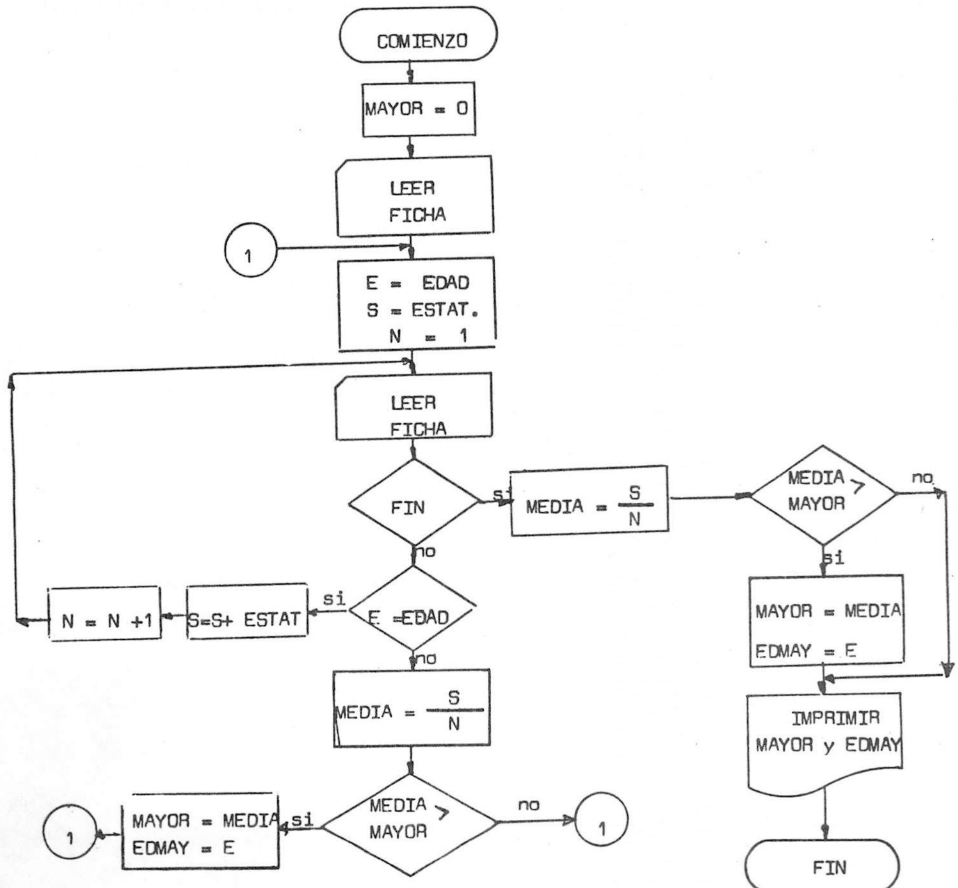
## PROBLEMA 65

Dado un fichero en tarjetas de una población, en el que existe un registro para cada individuo y en el que entre otros datos se detalla, la edad en años y la estatura en cm. Teniendo en cuenta que el fichero esta ordenado por edades se pide :

Edad de los individuos con mayor estatura mediA.

Esto es, calculando la estatura media de todos los individuos de una misma edad, se buscará la edad en la que esa media es mayor y se imprimirá la edad y la estatura media.

### RESOLUCION



## PROBLEMA 66

Sobre el archivo nuestro del archivo anterior se procesa un archivo de proceso en fichas que contiene exclusivamente nuevos suscriptores. No pueden nunca coincidir nº de suscriptor en cinta y nº de suscriptor en ficha.
Analizar el problema y hacer un ordinograma para resolverlo.

## PROBLEMA 67

Una empresa dispones en la memoria de su ordenador de una tabla multiple de tres entradas: nº de articulo, existencia del mismo, y stock mínimo.
Las operaciones de venta se registran en cinta con el formato, nº de artículo, cantidad vendida. Las operaciones de compra, en fichas con igual formato. Se qu8iere:

- Actualizar la tabla.
- Imprimir un listado en el que aparezcan el nº de artúculo y las existencias actualizadas, nº de operaciones de venta y nº de operaciones de compra..Debemos controlar el listado.

La tabla ni es coincidente ni esta ordenada, y tiene 100 elementos.
Analizar el problema y hacer un ordinograma para resolverlo.

# PROBLEMA 68

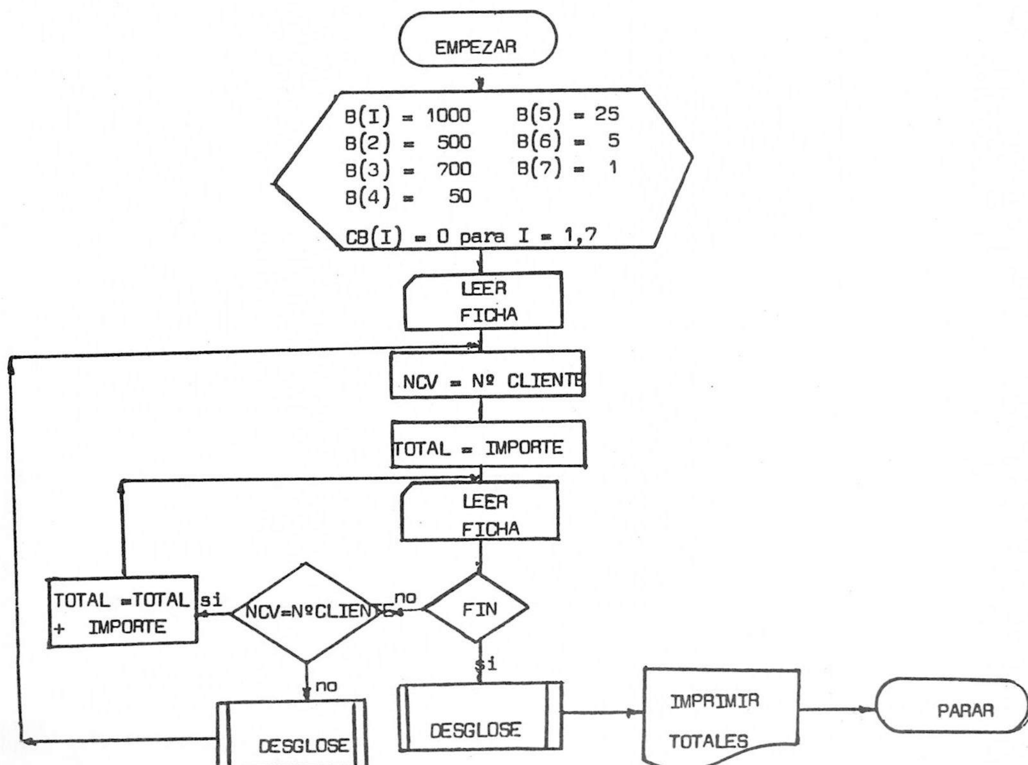
Se trata de hacer un organigrama que describa el desglose de moneda necesaria para el pago de unas facturas a proveedores.

El total a pagar a cada proveedor se deduce de unas fichas en cada unas de las cuales aparece un nº de cliente y un importe. Estas fichas están ordenadas por nº de cliente, pudiendo existir varias para cada cliente. Se pide pues, el desglose de moneda necesario para que sea posible el pago a todos los clientes.

## RESOLUCION

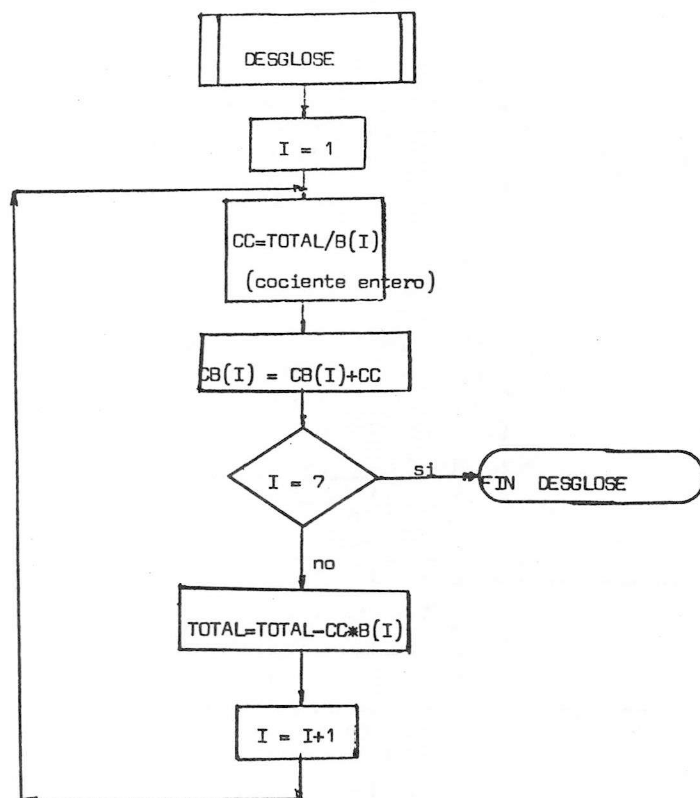
Para la resolución emplearemos dos tablas. La tabla B de siete elementos guarda los valores de las distintas monedas, esto es : 1000, 500, 100, 50, 25, 5, 1. La tabla CB tambien de 7 elementos guarda la cantidad de monedas(o billetes) de cada tipo que son necesarias.

Además utilizaremos una rutina para hacer el desglose propiamente dicho.




(Continua en la pagina siguiente)

(Viene de la pagina anterior)

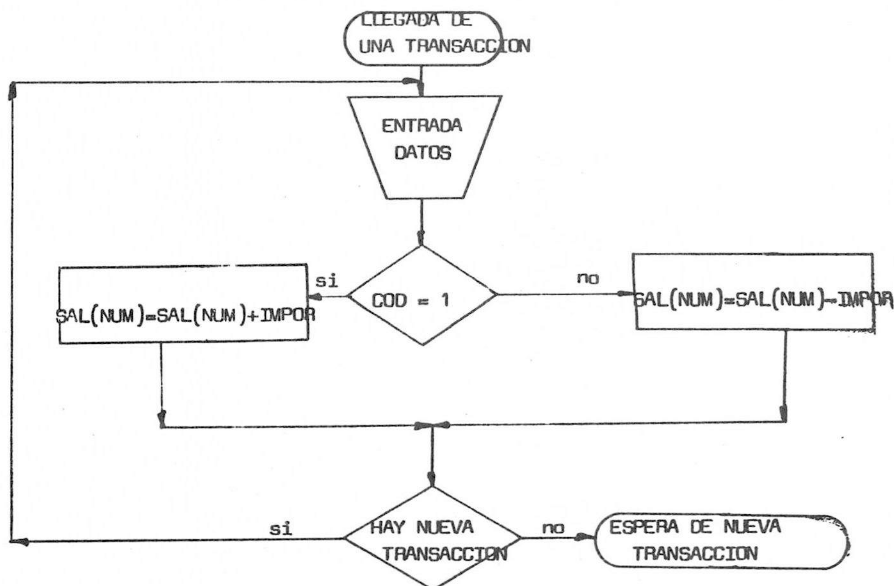


## PROBLEMA 69

Un banco tiene en memoria una tabla de saldos de cuentas ordenadas de forma que cada saldo ocupa en la tabla el lugar igual al nº de la cuenta correspondiente. A esta tabla la llamaremos SM.

Cada vez que se hace una operacción en una cuenta se introduce por un terminal el número de la cuenta y el importe de la transacción y un código(1, si es a favor del cliente y 2 en caso contrario ). Se pide un organigrama para tratar cada transacción. Esto es, actualizar el saldo correspondiente.

### RESOLUCION




Normalmente no es posible tener una tabla de este tipo ya que los números de cuentas suelen estar formados a base de claves mnemotécnicas etc... En este caso el problema se plantearía como el siguiente.

## PROBLEMA 70

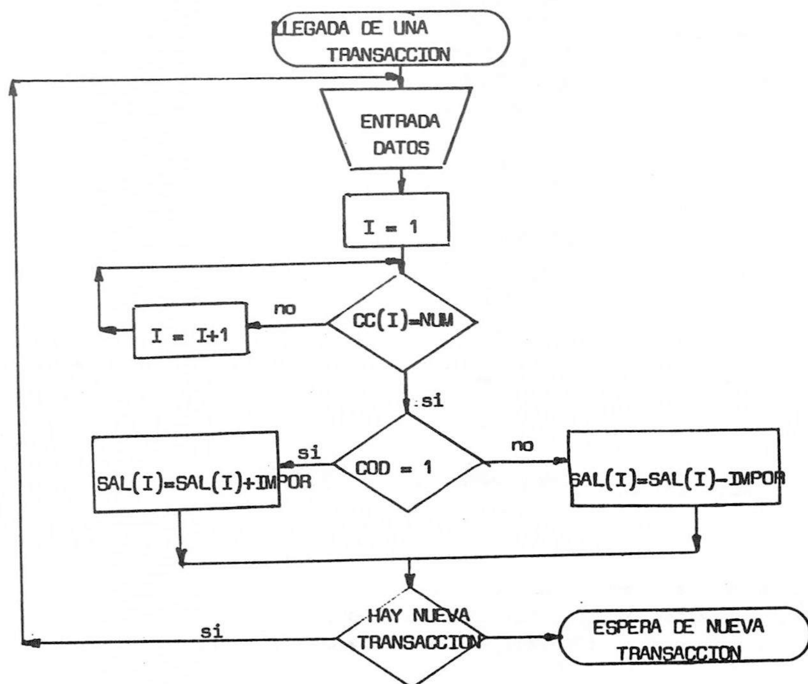
Un banco tiene en memoria una tabla doble en la que figuran por un lado los números de cuenta de los clientes y por otro los saldos de las correspondientes cuentas, que llamaremos CC y SAL.

Cada vez que se hace una operación en una cuenta se introduce por un terminal el número de la cuenta , el importe de la operación y un código(1 si es a favor ddel cliente y 2 en caso contrario).

Se pide un organigrama para tratar cada transacción.

### RESOLUCION

En este caso no podemos acceder directamente a los saldos sino que tendremos que buscar el lugar que ocupa este dentro de la tabla de saldos, mediante la tabla de números de cuentas.



## PROBLEMA 71

En una empresa se tiene un fichero en cinta magnética correspondiente a todas las facturas que debe cobrar esa empresa a fin de mes.

Cada registro corresponde a una factura y contiene :

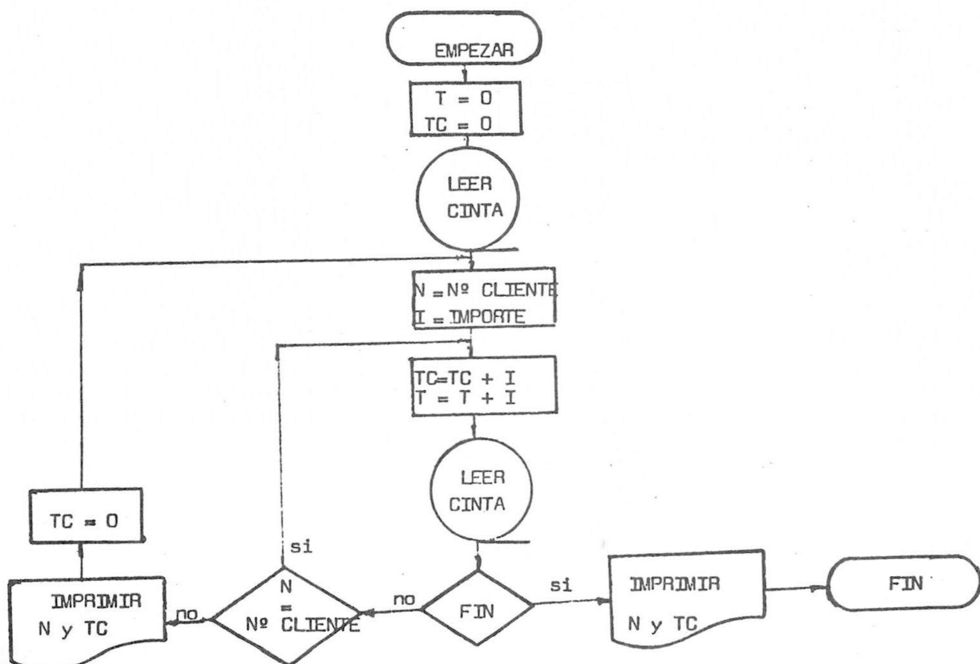
- nº de cliente.
- Importe a pagar.

Los registros están ordenados por nº de cliente y pueden existir varias tarjetas del mismo cliente.

Hacer un organigrama para obtener una relación de todos los clientes con el importe total que debe abonar cada uno. Al final la suma total de todas las facturas.

## RESOLUCION

Llevaremos por separado dos totales: TC el total del cliente que se esté procesando y T el total de todos.



## PROBLEMA 72

En una empresa existe un archivo en cinta, maestro de empleados, en el que existen entre otros datos los siguientes :

Clave del empleado ...................... CLAVE
Nombre y Apellidos ...................... NOMB
Sueldo mensual ...................... S
Estado civil ...................... EC
Número de hijos ...................... NH
Categoria ...................... C

Este archivo se actualiza con un archivo de movimientos en tarjeta con la siguiente información por registro :

Clave de empleado ...................... CLAVEM
Nombre y Apellidos ...................... NOMBM
Sueldo modificado ...................... SM
Estado civil modificado ................ ECM
Número de hijos modificado ............ NHM
Categoria modificada .................. CM

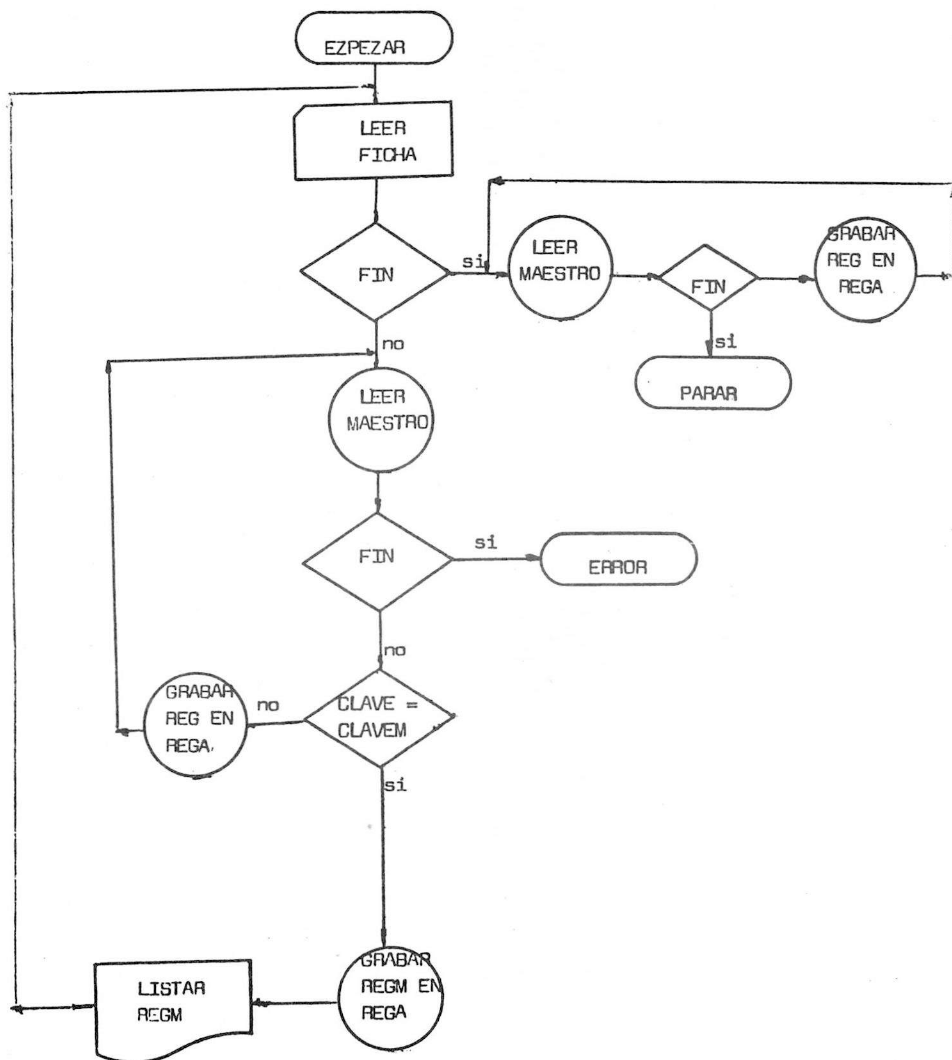
Si alguno de los datos no varia se perforará el mismo. Se trata de hacer una actualización del archivo maestro con el archivo de movimientos, debiendose listar todas las modificaciones.

NOTA : Se supone que no hay altas ni bajas.

RESOLUCION

Al registro del maestro le llamamos REG al actualizado REGA y al de movimientos REGM.

(La solución en la pagina siguiente)


EZPEZAR
LEER
FICHA
FIN
si
no
LEER
MAESTRO
FIN
si
GRABAR
REG EN
REGA
PARAR
LEER
MAESTRO
FIN
si
ERROR
no
GRABAR
REG EN
REGA.
no
CLAVE =
CLAVEM
si
GRABAR
REGM EN
REGA
LISTAR
REGM

## PROBLEMA 73

Se trata de realizar la actualización de un fichero en cinta magnética. El fichero a actualizar contiene un registro por indicativo y está clasificado por orden creciente de indicativo.

Las modificaciones están en un fichero de movimientos en tarjetas perforadas, cada una de las cuales tiene un código que indica la operación.

A = Indica alta.
B = Indica baja.
M = Indica modificación.

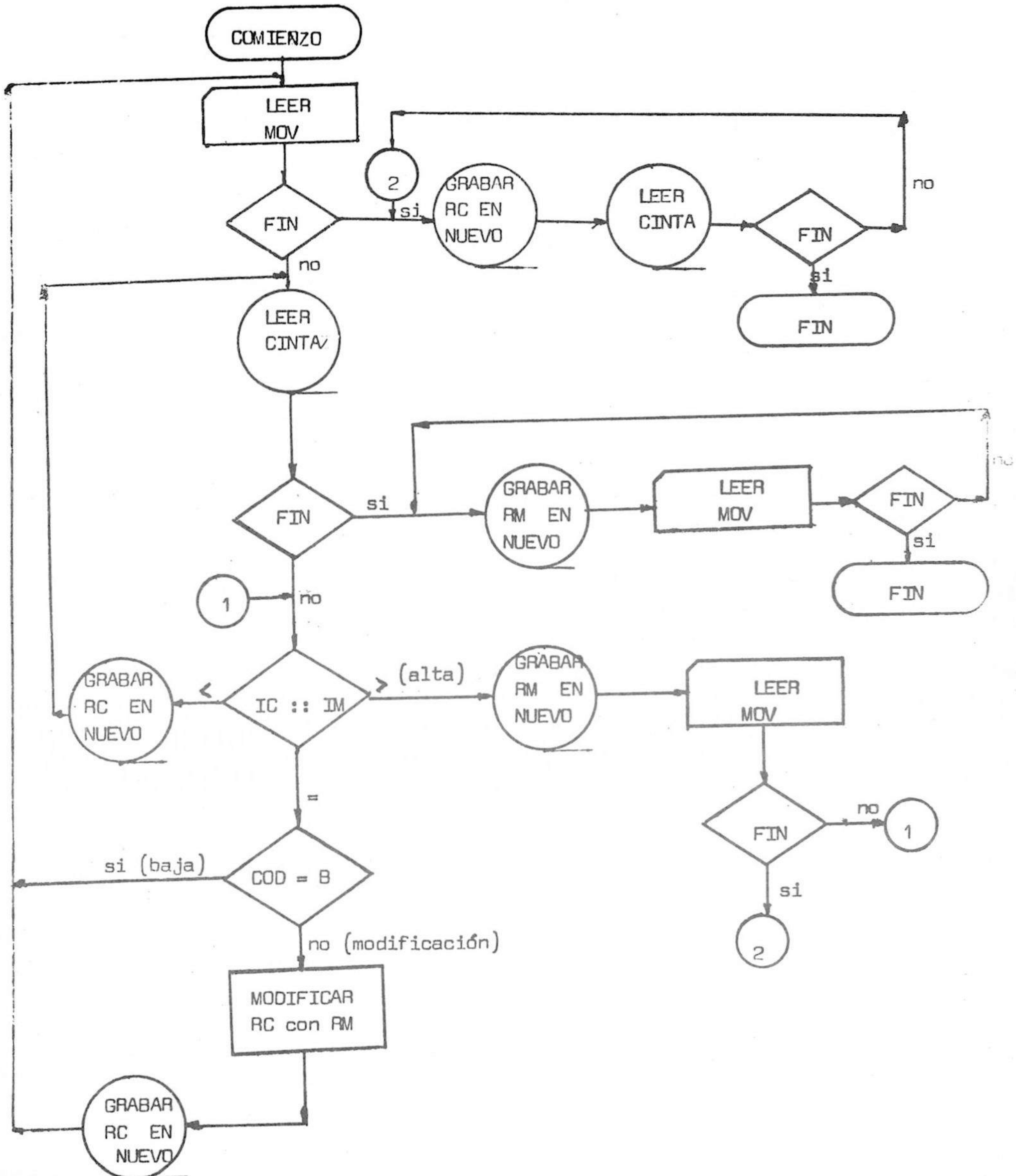
El fichero en tarjetas está clasificado en orden creciente de indicativo. Admitiremos que no hay ningún error ni en el archivo maestro ni en el de movimientos.

### RESOLUCION

Utilizaremos los siguientes nombres :

Archivo a actualizar = CINTA
Registro de CINTA = RC
Archivo de movimientos = MOV
Registro de MOV = RM
Archivo actualizado = NUEVO
Registro de NUEVO = RN
Indicativo de RC = IC
Indicativo de RM = IM

(Resolución en la pagina siguiente)


COMIENZO
LEER MOV
FIN
si
no
2
GRABAR RC EN NUEVO
LEER CINTA
FIN
no
si
FIN
LEER CINTA
FIN
si
GRABAR RM EN NUEVO
LEER MOV
FIN
no
si
FIN
1
no
GRABAR RC EN NUEVO
<
IC :: IM
>
(alta)
GRABAR RM EN NUEVO
LEER MOV
FIN
no
1
si
2
=
si (baja)
COD = B
no (modificación)
MODIFICAR RC con RM
GRABAR RC EN NUEVO

## PROBLEMA 74

Una revista tiene en cinta un archivo de suscriptores. Cada registro tiene el número de suscriptor y la fecha en que expira. Los registros están ordenados por número de suscriptor. Se perforan en fichas las renovaciones de suscripción. Cada ficha contiene el nº de suscriptor y la nueva fecha de expiración. Estas fichas se ordenan por nº de suscriptor y se coloca como primera ficha, una que contiene la fecha actual.

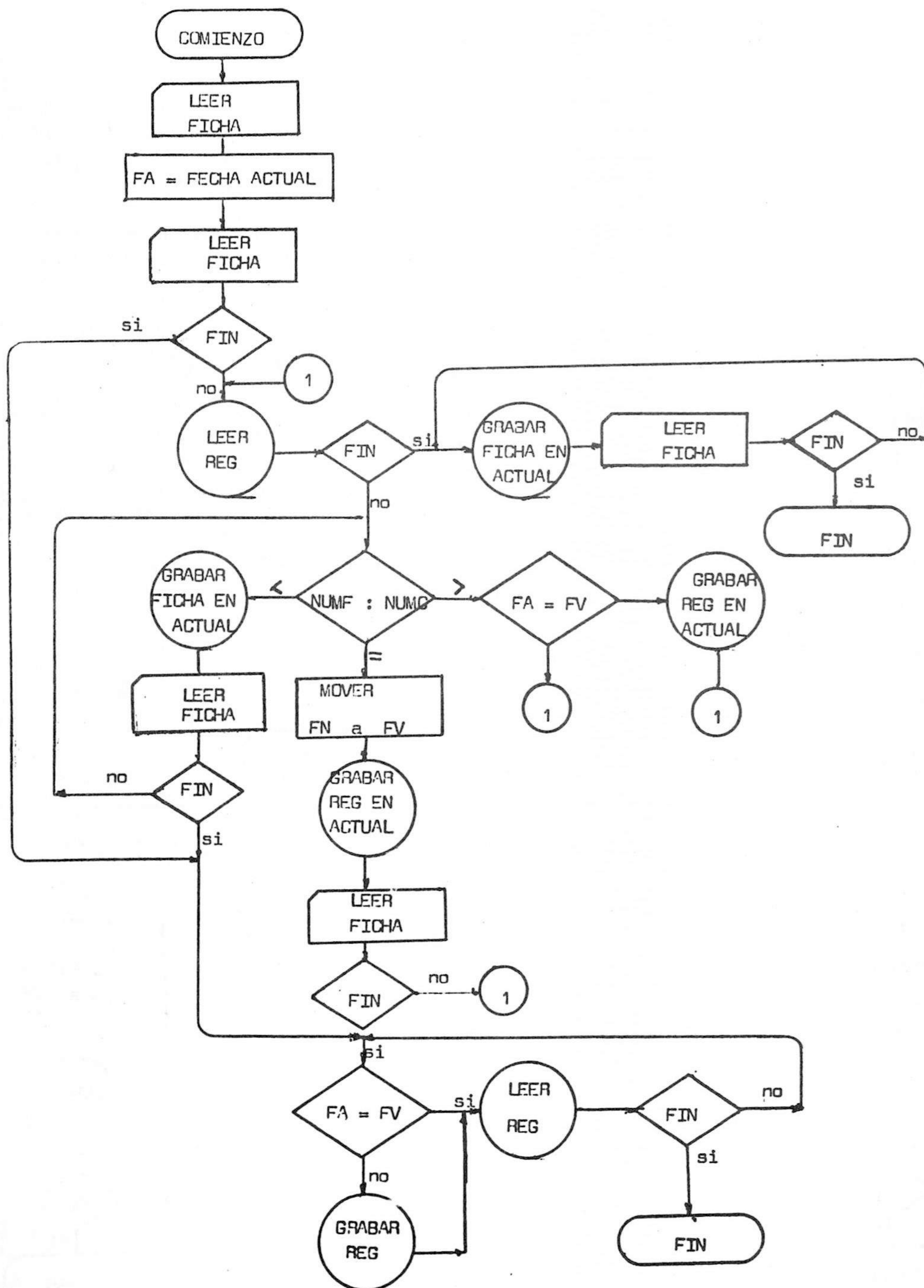
Se pide un organigrama para actualizar el archivo de suscriptores. Esto es, incluir las altas; dar de baja a aquellos suscriptores que habiendo llegado la fecha de expiración de su suscripción no la han renovado; y modificar la fecha de aquellos que si han renovado su suscripción.

### RESOLUCION

Utilizaremos los siguientes nombres :

| | | |
|---|---|---|
| Archivo de suscriptores | : | CINTA |
| Registro de CINTA | : | REG |
| Archivo de movimientos | : | FICHERO |
| Registro de FICHERO | : | FICHA |
| Archivo actualizado | : | ACTUAL |
| Nº cliente de REG | : | NUMC |
| Nº cliente de FICHA | : | NUMF |
| Fecha de REG | : | FV |
| Fecha de FICHA | : | FN |
| Fecha actual | : | FA |

(Solución en la página siguiente)



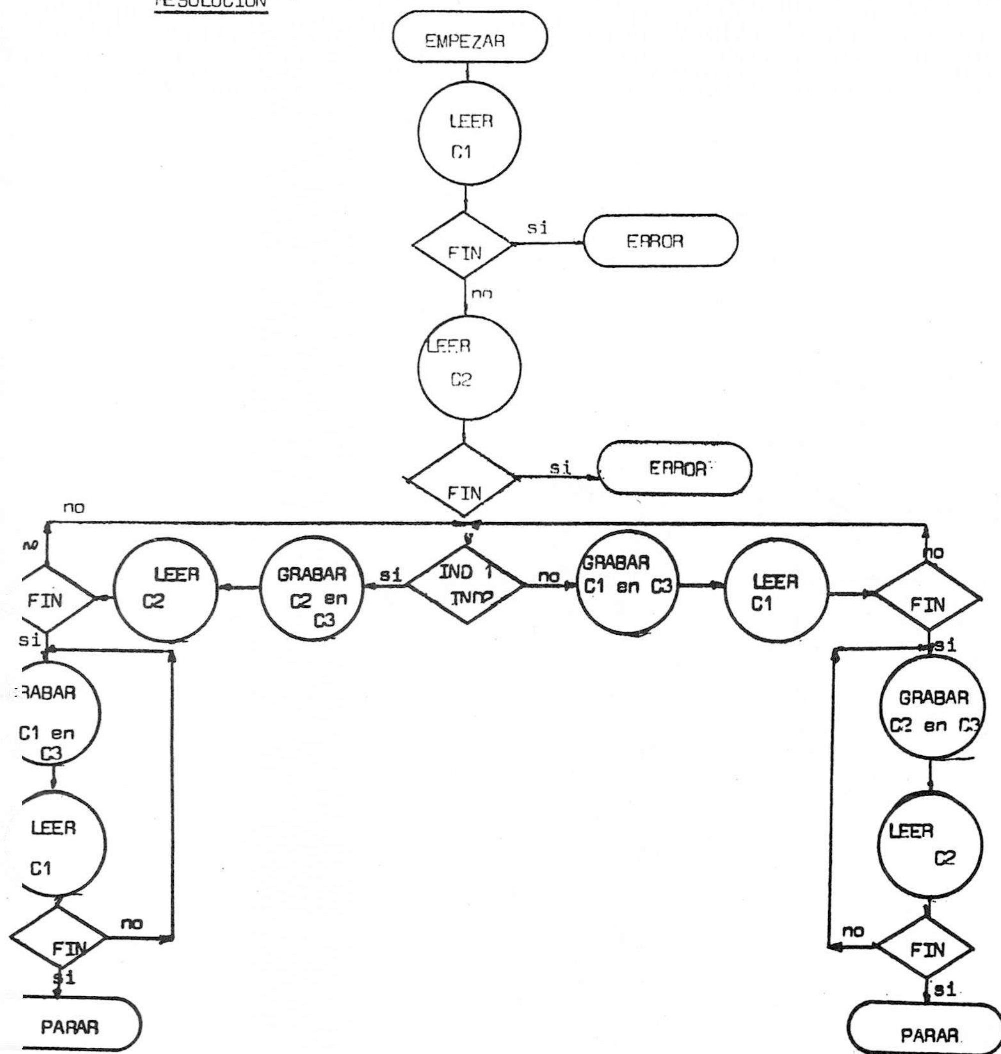
COMIENZO
LEER FICHA
FA = FECHA ACTUAL
LEER FICHA
FIN
si
no
1
LEER REG
FIN
si
GRABAR FICHA EN ACTUAL
LEER FICHA
FIN
no
si
FIN
no
GRABAR FICHA EN ACTUAL
<
NUMF : NUMC
>
FA = FV
GRABAR REG EN ACTUAL
=
LEER FICHA
MOVER FN a FV
1
1
no
FIN
GRABAR REG EN ACTUAL
si
LEER FICHA
FIN
no
1
si
FA = FV
si
LEER REG
FIN
no
no
si
GRABAR REG
FIN

PROBLEMA **75**

Se trata de obtener una cinta que resulte de la intercalación de otras dos. En ambas cintas los registros son de longitud fija y cada uno contienen un indicativo. Los registros están ordenados , en las dos cintas, en orden creciente de indicativo y se supone que no existen dos registros con el mismo indicativo.

Describir mediante un organigrama el proceso necesario para obtener una tercera cinta en la que exist n todos los registros de las otras dos ordenados por orden creciente.

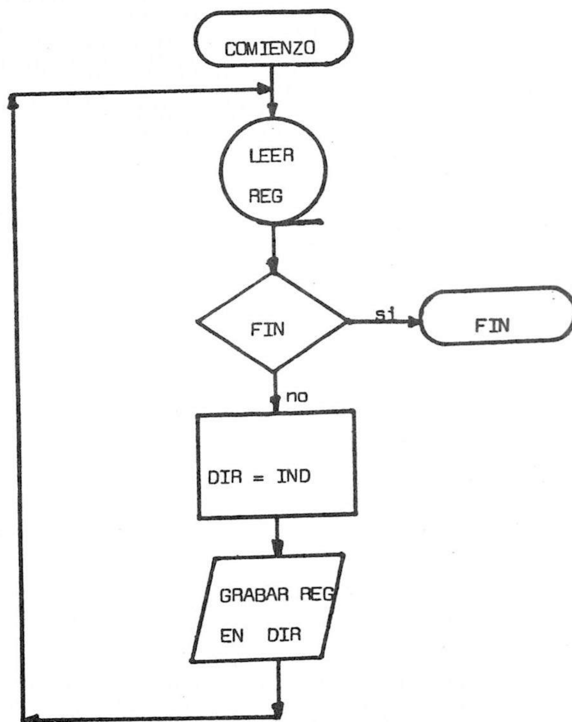
RESOLUCION



## PROBLEMA 76

Dado un archivo en cinta cuyos registros tienen un indicativo IND de 3 dígitos, se desea grabarlo en un disco para poder consultarlo directamente, utilizando como dirección el propio indicativo. Se supone que no existen indicativos repetidos.

RESOLUCION

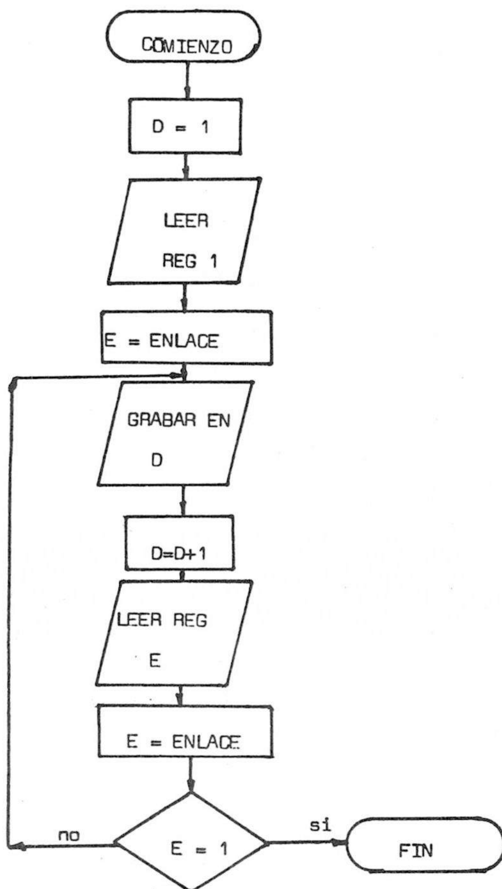


## PROBLEMA 77

Se sabe que cuando en un fichero con organización secuencial encadenado, se han realizado ya un número grande de inserciones, la consulta al mismo puede ser muy larga comparada con la de un fichero puramente secuencial, debido a los accesos a la zona secundaria. Se dice entonces que el fichero se ha degenerado.

Se pide un organigrama para regenerar un fichero secuencial encadenado degenerado. Suponemos que el primer registro está en la dirección 1 del archivo, y que el último registro está encadenado al primero.

### RESOLUCION



# PROBLEMA 78

El banco "Dolares y Sufrimiento" tiene en cinta magnetica un archivo de cuentas corrientes. Cada registro contiene nº de cuenta y saldo de su cuenta. Los registros están ordenados en secuencia creciente por el número de cuenta.
Semanalmente se perfora en ficha las operaciones (ingresos y pagos) sobre las cuentas. Cada ficha contiene nº de cuenta, importe de la operación, un código. El código puede ser: I= Ingresos; P= Pagos. Cada cliente puede tener varias fichas.
Se procesan las fichas ordenadas por el nº de cuenta contra el archivo maestro para obtener en otra cinta un archivo maestro actualizado. Si el saldo actualizado de un registro es negativo, se imprime dicho registro en un listado sin controlar.
Todas las fichas tienen registro.
Analizar el problema y hacer un ordinograma para resolverlo.

# PROBLEMA 79

Se trata de crear un fichero en un soporte direcionable que puede ser consultado mediante acceso directo.

Los registros vienen en fichas con un indicativo de la forma A999. Se supone que no existen registros con el mismo indicativo.

Hay que crear el fichero de forma que se pueda acceder directamente a cada registro.

## RESOLUCION

La dirección la obtendremos directamente del indicativo, de la forma siguiente: La dirección será un número de 5 dígitos, los tres últimos los mismos del indicativo y los dos primeros los obtendremos transformando la letra en un número de dos dígitos. Para facilitar el proceso crearemos en memoria una tabla A de 26 elementos en la que cada elemento contiene una letra, y buscaremos en ella cada letra, siendo el lugar que ocupa el valor que asignamos a la letra.

El indicativo tendrá la forma siguiente :

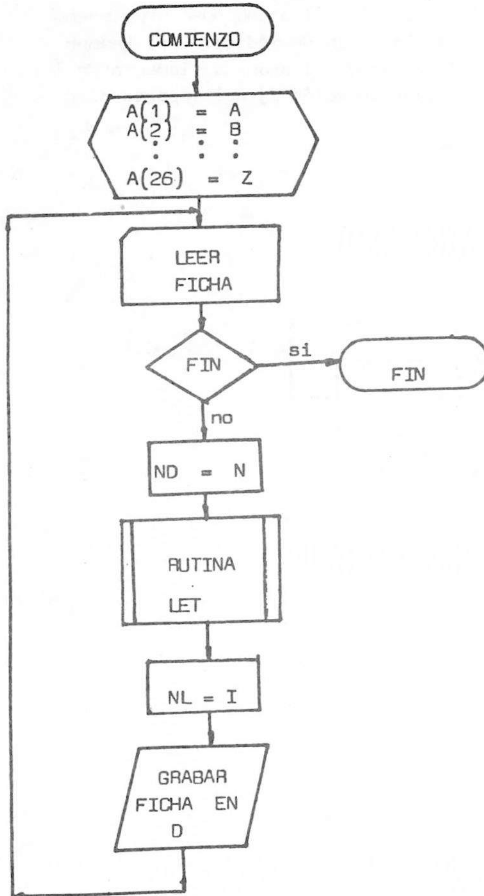


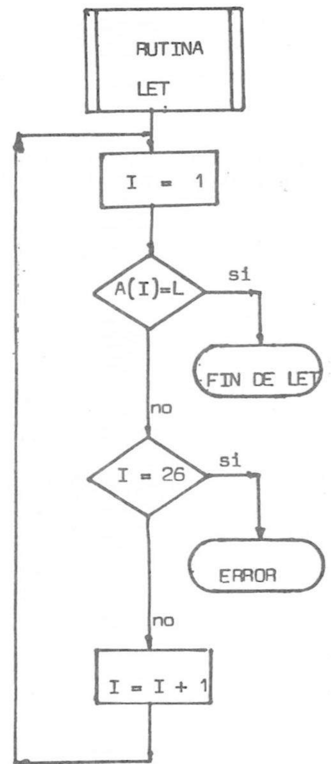

La dirección tendrá la forma siguiente :




Crearemos una rutina que llamaremos LET, para transformar una letra en un número.

(Solución en la pagina siguiente)


COMIENZO
A(1) = A
A(2) = B
A(26) = Z
LEER
FICHA
FIN
si
FIN
no
ND = N
RUTINA
LET
NL = I
GRABAR
FICHA EN
D


RUTINA
LET
I = 1
A(I)=L
si
FIN DE LET
no
I = 26
si
ERROR
no
I = I + 1

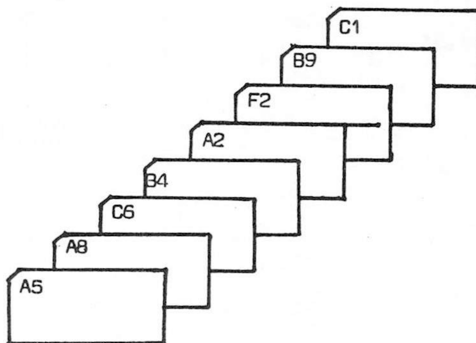
## PROBLEMA 80

Se desea crear un archivo en un soporte direccionable para ser consultado mediante acceso directo. Los registros a crear vienen en unas fichas desordenadas. Para obtener la dirección de cada registro se creará una tabla de indicativos y operaremos de la forma siguiente :

Se lee el primer registro(la primera ficha), y se graba en la primera dirección del soporte direccionable(dirección 1), almacenandose previamente su indicativo en el primer elemento de la tabla de indicativos. Leemos el segundo registro, almacenando su indicativo en el segundo elemento de la tabla, y grabandolo en la segunda dirección(dirección 2) del soporte direccionable, y así sucesivamente.

Un ejemplo aclarará más el problema :

FICHAS




SOPORTE DIRECCIONABLE

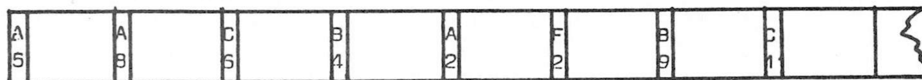


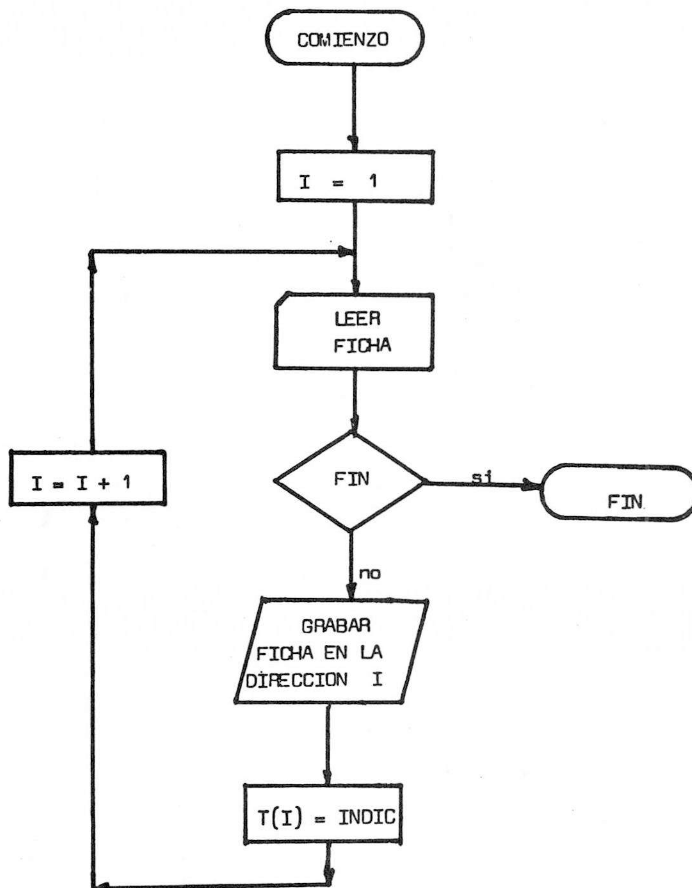

TABLA DE MEMORIA

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A5 | A8 | C6 | B4 | A2 | F2 | B9 | C1 |

Como vemos los registros se van grabando en el soporte direccionable en el orden en que se obtienen, pero no existe ningún orden ni relación entre sus indicativos. Para poder luego acceder directamente a un registro, vamos guardando el indicativo de cada registro en la tabla, y de esta forma, buscando en la tabla, sabremos que lugar ocupa el registro en el soporte direccionable, esto es, en que dirección está.

Se pide, un organigrama para crear este archivo y la correspondiente tabla.

RESOLUCION



## PROBLEMA 81

Una vez terminado el proceso del ejercicio anterior, tendremos el archivo grabado en un soporte direccionable y la tabla de indicativos en memoria. Se desea ahora consultar el archivo. La consulta consistirá en encontrar y listar determinados registros, conocidos sus indicativos. Los indicativos de los registros a consultar vienen en tarjetas perforadas a razón de un indicativo por tarjeta.

Se pide un organigrama que describa el procedimiento de consulta a partir de estas fichas.

RESOLUCION

